

易语言

入门与提高

霍玲玲 徐文军 张晓华 王成武
赵云青 镇 维 张志军 编著

易语言——全中文的编程语言

- 全中文支持，中文本土文化特色。
- 全部自主知识产权。
- 内置专用输入法，支持中文语句快速录入。
- 多种语言支持：简体中文版、繁体中文版、英文版、日文版。
- 全可视化编程，界面设计及程序流程即时可视。
- 独立的高质量编译器，源程序直接编译为目的机器的CPU指令。
- 跨平台编程，支持Windows和Linux，不依赖特定操作系统。
- 拥有自己的数据库系统，支持访问现有所有数据库。
- 完善的网络、端口通信和互联网功能支持。
- 强大的多媒体功能支持。



随书附光盘一张



国防工业出版社
National Defense Industry Press

责任编辑：杨星豪 xhyang@ndip.cn
责任校对：钱辉玲
封面设计：蒋秀芹

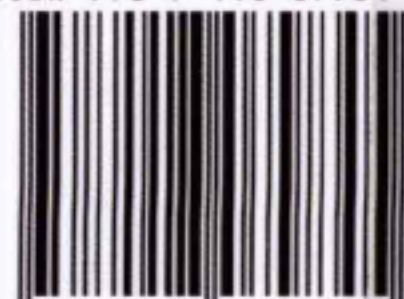
易语言

入门与提高

► 上架建议：程序设计/计算机语言/计算机/电子信息 ◀

<http://www.ndip.cn>

ISBN 978-7-118-07984-5



9 787118 079845 >

定价：56.00 元（含光盘）

易语言入门与提高

霍玲玲 徐文军 张晓华 王成武 编著
赵云青 镇 维 张志军

国防工业出版社

· 北京 ·

内 容 简 介

本书从初学者的角度出发,全面介绍了易语言的相关知识和基础内容。全书共分为5个部分:第一部分(第1、2章)介绍了易语言的基础知识,包括易语言的特点;易语言的下载与安装;易语言基本界面操作;易语言的变量、常量、运算符及表达式的使用等;第二部分(第3、4章)介绍了易语言的命令,包括基本命令(算术运算、逻辑比较、数组操作等)和常用命令(时间操作、磁盘操作、文件读写、系统处理等);第三部分(第5~11章)介绍了易语言核心支持库中组件的使用。包括各组件的属性、方法、事件等;第四部分(第12、13章)介绍了易语言的数据库库组件和网络组件;第五部分(第14~16章)介绍了如何编写易语言模块、DLL及如何调用模块、APP、DLL等。

本书附赠光盘1张,包括书中引用的每个例程源代码。

本书适合没有编程基础的易语言初学者作为入门教程,或作为喜好编程且想尽快上手的编程爱好者的启蒙教程,也可作为大、中专院校和培训班的教材,对于易语言开发的爱好者,本书也有较大的参考价值。

图书在版编目(CIP)数据

易语言入门与提高 / 霍玲玲等编著. —北京:国防工业出版社, 2012.4
ISBN 978-7-118-07984-5

I. ①易… II. ①霍… III. ①汉语—程序语言—程序设计 IV. ①TP312

中国版本图书馆CIP数据核字(2012)第051596号

※

国防工业出版社出版发行
(北京市海淀区紫竹院南路23号 邮政编码100048)
北京奥鑫印刷厂印刷
新华书店经售

*

开本 787×1092 1/16 印张 26 字数 605 千字

2012年4月第1版第1次印刷 印数 1—4000册 定价 56.00元(含光盘)

(本书如有印装错误,我社负责调换)

国防书店:(010)88540777

发行邮购:(010)88540776

发行传真:(010)88540755

发行业务:(010)88540717

前 言

易语言是一款全中文全可视化跨平台的编程工具,以中文作为程序代码表达的语言形式,由大连大有吴涛易语言软件开发有限公司设计开发,以“易”著称。易语言作为一款全中文的编程语言,将中华文化和民俗习惯彻底融入了计算机语言领域,用户不再需要被强制学习和掌握国外的文化习惯、表达方式。用户只要会使用计算机和文字输入就可以进行程序设计。

易语言早期版本的名字为E语言,最早版本的发布可追溯至2000年9月11日。从2000年至今,易语言已经发展到一定的规模:功能上、用户数量上都十分可观。易语言如此快速地发展,主要得益于它所具有的强大功能:第一,易语言是全中文支持,全部自主知识产权,和其他国外语言相比,易语言最大的不同是彻底中文化,且拥有自下而上的全部自主知识产权;第二,易语言拥有自己独立的高质量编译器,源程序被直接编译为目的机器的CPU指令,不存在任何速度瓶颈和安全隐患;第三,易语言可跨平台编程,现已同时支持Windows和Linux程序开发,不再依赖特定的操作系统,为国家推广Linux操作系统提供应用软件开发工具;第四,易语言已完全解决了输入速度的问题,中文语句的输入速度绝对不亚于英文语句输入速度;第五,易语言拥有自己的数据库系统,且支持访问现有所有数据库。

本书适合没有编程基础的初学者作为入门教程,主要特点有:本书中引用的概念准确,必要时进行类比,读者很容易理解;本书由浅入深,循序渐进的科学编排模式,读者很容易掌握;全书提供了大量实例,每个实例涵盖了易语言的一个知识点,既有反映知识要点的小实例,也有较大的综合实例,所有实例中用到的源代码都放在本书的配套光盘中供读者学习使用;本书对代码进行了丰富的注释,阅读起来没有任何障碍;本书中无论理论知识还是实例讲解都很详细,很容易理解。

全书分为5个部分:第一部分(第1、2章)介绍了易语言的基础知识,包括易语言的特点;易语言的下载与安装;易语言基本界面操作;易语言的变量、常量、运算符及表达式的使用等;第二部分(第3、4章)介绍了易语言的命令。包括基本命令(算术运算、逻辑比较、数组操作等)和常用命令(时间操作、磁盘操作、文件读写、系统处理等);第三部分(第5~11章)介绍了易语言核心支持库中组件的使用。包括各组件的属性、方法、事件等;第四部分(第12、13章)介绍了易语言的数据库库组件和网络组件;第五部分(第14~17章)介绍了如何编写易语言模块、DLL;及如何调用模块、APP、DLL等。

本书由赵云青、镇维统稿,第1、2章由镇维编写,第3、7章由王成武编写,第4、5章由张志军编写,第6、13章由赵云青编写,第8、9章由霍玲玲编写,第10、11、12章由张晓华编写,第14、15、16由徐文军编写。另外,张景生高工给予了本书原则指导。

部分例程参考了网上易语言爱好者代码,在此表示感谢。由于编者水平有限,书中也难免存在谬误,恳请读者批评指正。

目 录

第1章 易语言概述	1		
1.1 易语言概要	1		
1.2 易语言的下载与安装	17		
1.3 易语言基本界面操作	20		
1.3.1 易语言的界面	20		
1.3.2 易语言的菜单栏	23		
1.3.3 易语言的工具条	34		
1.3.4 如何在设计窗口中 添加组件	35		
1.3.5 如何使用易语言帮助系统 ...	35		
1.3.6 如何配置易语言	35		
1.4 易语言代码输入方法	37		
1.4.1 内置输入法	37		
1.4.2 系统输入法	38		
1.4.3 参数分步输入	38		
1.4.4 输入备注与代码屏蔽	38		
1.4.5 4种输入语句	39		
第2章 易语言基础知识	40		
2.1 易语言数据类型	40		
2.2 变量与常量	42		
2.2.1 变量	42		
2.2.2 常量	45		
2.3 易语言运算符及表达式	48		
2.3.1 易语言运算符	48		
2.3.2 算术运算符和算术表达式 ...	49		
2.3.3 赋值运算符和赋值表达式 ...	50		
2.4 易语言命令介绍	51		
2.4.1 命令的格式	51		
2.4.2 即时帮助和帮助文档	51		
2.4.3 命令的返回值	52		
2.4.4 命令的套用	53		
2.5 易语言组件介绍	54		
2.5.1 基本组件	54		
2.5.2 扩展组件	55		
2.6 成员属性、方法与事件	56		
2.6.1 通用属性	56		
2.6.2 通用方法	56		
2.6.3 成员事件	57		
第3章 易语言基本命令	58		
3.1 流程控制	58		
3.1.1 分支类流程控制命令	58		
3.1.2 循环类流程控制命令	61		
3.1.3 跳转类流程控制命令	63		
3.2 算术运算	65		
3.2.1 基本算术运算命令	65		
3.2.2 扩展算术运算命令	66		
3.3 逻辑比较与位运算	69		
3.3.1 逻辑比较	69		
3.3.2 位运算	71		
3.4 数组操作	74		
3.5 数值转换	80		
第4章 易语言常用命令	86		
4.1 时间操作	86		
4.2 文本操作与字节集操作	89		
4.2.1 文字编码和存储方式	89		
4.2.2 ASCII 码	89		
4.2.3 区别键代码和文字编码	90		
4.2.4 文本操作命令	90		
4.2.5 字节集操作命令	92		
4.3 磁盘操作	92		
4.3.1 目录路径	93		
4.3.2 磁盘操作命令	93		
4.4 文件读写	94		
4.5 媒体播放	96		

4.5.1 媒体播放格式	96	6.2.2 图形按钮的属性	155
4.5.2 媒体播放命令	96	6.2.3 图形按钮的事件	157
4.6 系统处理	97	6.3 调节器组件	158
4.6.1 了解剪切板	97	6.3.1 调节器组件概述	158
4.6.2 了解注册表	97	6.3.2 调节器的属性	159
4.6.3 系统处理命令	98	6.3.3 调节器的重要事件	159
第5章 主体组件	105	6.3.4 与编辑框调节器的对比 ...	160
5.1 窗口组件	105	6.4 拖放对象组件	161
5.1.1 窗口概述	105	6.4.1 拖放对象组件概述	161
5.1.2 窗口的属性	107	6.4.2 拖放对象组件重要属性 ...	161
5.1.3 窗口的事件	115	6.4.3 拖放对象组件的方法	162
5.1.4 窗口的方法	122	6.4.4 拖放对象组件的事件	162
5.2 菜单组件	130	第7章 文本与图形组件	164
5.2.1 菜单概述	131	7.1 编辑框组件	164
5.2.2 菜单的热键与属性	133	7.1.1 编辑框概述	164
5.2.3 弹出菜单	135	7.1.2 编辑框的属性	164
5.2.4 托盘菜单	137	7.1.3 编辑框的方法	170
5.3 超级菜单	138	7.1.4 编辑框的事件	171
5.3.1 超级菜单的属性	139	7.2 标签组件	174
5.3.2 超级菜单的方法	140	7.2.1 标签概述	174
5.3.3 超级菜单的事件	140	7.2.2 标签属性	175
5.4 信息框组件	141	7.2.3 标签的应用例程	177
5.4.1 信息框概述	141	7.3 画板组件	181
5.4.2 信息框提示按钮形态	141	7.3.1 画板概述	181
5.4.3 信息框的返回值	143	7.3.2 画板的属性	181
5.5 输入框组件	143	7.3.3 画板的方法	182
5.5.1 输入框操作	144	7.3.4 画板的事件	186
5.5.2 输入框初始数据	145	7.4 图片框组件	187
5.5.3 输入框输入编程	146	7.4.1 图片框组件概述	187
5.6 数据库维护	147	7.4.2 图片框组件的重要属性 ...	187
5.7 浏览文件夹	147	7.5 外形框组件	189
第6章 鼠标触发组件	149	7.5.1 外形框组件概述	189
6.1 按钮组件	149	7.5.2 外形框组件属性	189
6.1.1 按钮组件概述	149	7.6 影像框组件	192
6.1.2 按钮的重要属性	149	7.6.1 影像框组件概述	192
6.1.3 按钮的专有方法	152	7.6.2 影像框组件属性	192
6.1.4 按钮的重要事件	152	7.7 柱状图组件、饼形图组件和	
6.1.5 按钮的提示信息	154	曲线图组件	193
6.2 图形按钮	155	7.7.1 柱状图组件、饼形图组件和	
6.2.1 图形按钮概述	155	曲线图组件概述	193

7.7.2 柱状图组件、饼形图组件和 曲线图组件属性	193	8.9.2 选择列表框的属性	234
第8章 选择与列表组件	196	8.9.3 选择列表框的方法	235
8.1 选择框组件	196	8.9.4 选择列表框的事件	237
8.1.1 选择框概述	196	第9章 进度与时间组件	239
8.1.2 选择框的属性	196	9.1 进度条组件	239
8.1.3 选择框的事件	197	9.1.1 进度条概述	239
8.2 单选框组件	198	9.1.2 进度条的属性	239
8.2.1 单选框概述	198	9.1.3 进度条的事件	240
8.2.2 单选框的属性	199	9.2 滑块条组件	241
8.2.3 单选框的事件	200	9.2.1 滑块条概述	241
8.3 分组框组件	200	9.2.2 滑块条的属性	241
8.3.1 分组框概述	200	9.2.3 滑块条的事件	242
8.3.2 分组框的属性	201	9.3 横、纵向滚动条组件	244
8.3.3 分组框的事件	201	9.3.1 横、纵向滚动条概述	244
8.4 选择夹组件	202	9.3.2 横、纵向滚动条的属性	244
8.4.1 选择夹概述	202	9.3.3 横、纵向滚动条的事件	245
8.4.2 选择夹的属性	202	9.4 日期框组件	246
8.4.3 选择夹的方法	204	9.4.1 日期框概述	246
8.4.4 选择夹的事件	205	9.4.2 日期框的属性	246
8.5 分隔条组件	206	9.4.3 日期框的事件	247
8.5.1 分隔条概述	206	9.5 月历组件	247
8.5.2 分隔条的属性	206	9.5.1 月历概述	247
8.5.3 分隔条的事件	206	9.5.2 月历的属性	247
8.6 通用对话框组件	207	9.5.3 月历的事件	249
8.6.1 通用对话框概述	207	9.6 农历日期框组件	250
8.6.2 通用对话框的属性	207	9.6.1 农历日期框概述	250
8.6.3 通用对话框的方法	214	9.6.2 农历日期框的属性	250
8.7 列表框组件	214	9.6.3 农历日期框的方法	252
8.7.1 列表框概述	214	9.6.4 农历日期框的事件	252
8.7.2 列表框的属性	215	9.7 时钟组件	253
8.7.3 列表框的方法	216	9.7.1 时钟概述	253
8.7.4 列表框的事件	225	9.7.2 时钟的属性	254
8.8 组合框组件	226	9.7.3 时钟的事件	254
8.8.1 组合框概述	226	第10章 设备组件	256
8.8.2 组合框的属性	226	10.1 颜色选择器	256
8.8.3 组合框的方法	228	10.1.1 颜色选择器组件的属性	256
8.8.4 组合框的事件	232	10.1.2 颜色选择器组件的事件	256
8.9 选择列表框组件	234	10.2 打印机组件	258
8.9.1 选择列表框概述	234	10.2.1 打印机组件的属性	258
		10.2.2 打印机组件的方法	259

10.3 驱动器框组件	261	12.2 数据源组件	313
10.3.1 驱动器框组件的属性	262	12.2.1 数据源组件的属性	313
10.3.2 驱动器框组件的事件	262	12.2.2 数据源组件的事件	314
10.4 目录框组件	264	12.2.3 数据源组件的方法	314
10.4.1 目录框组件的属性	264	12.3 表格组件	316
10.4.2 目录框组件的事件	264	12.3.1 表格组件的属性	316
10.5 文件框组件	265	12.3.2 表格组件的事件	318
10.5.1 文件框组件的属性	266	12.3.3 表格组件的方法	318
10.5.2 文件框组件的事件	266	12.4 外部数据库	321
10.6 端口组件	268	12.4.1 外部数据库组件的属性 ...	322
10.6.1 端口组件的属性	268	12.4.2 外部数据库组件的方法 ...	322
10.6.2 端口组件的事件	270	12.5 外部数据提供者	328
10.6.3 端口组件的方法	270	第13章 网络组件	330
10.7 语音识别组件	272	13.1 数据报	330
10.7.1 语音识别组件的方法	272	13.1.1 数据报概述	330
10.7.2 语音识别组件的事件	273	13.1.2 数据报的重要属性	330
10.8 电话控制组件	274	13.1.3 数据报的专有方法	331
10.8.1 电话控制组件的属性	274	13.1.4 数据报的重要事件	331
10.8.2 电话控制组件的方法	274	13.2 客户/服务器组件	332
10.8.3 电话控制组件的事件	277	13.2.1 客户/服务器组件概述 ...	332
第11章 图片组组长	281	13.2.2 客户组件的重要属性	333
11.1 工具条组件	281	13.2.3 客户组件的专有方法	333
11.1.1 工具条组件的属性	281	13.2.4 客户组件的重要事件	334
11.1.2 工具条组件的事件	284	13.2.5 服务器的重要属性	334
11.1.3 工具条组件的方法	285	13.2.6 服务器的专有方法	334
11.2 状态条组件	288	13.2.7 服务器的重要事件	335
11.2.1 状态条组件的属性	289	13.3 超级链接框	335
11.2.2 状态条组件的事件	289	13.3.1 超级链接框概述	335
11.2.3 状态条组件的方法	290	13.3.2 超级链接框的重要属性 ...	336
11.3 树形框组件	292	13.3.3 超级链接框的专有方法 ...	337
11.3.1 树形框组件的属性	292	13.4 超文本浏览框组件	339
11.3.2 树形框组件的事件	294	13.4.1 超文本浏览框组件概述 ...	339
11.3.3 树形框组件的方法	295	13.4.2 超文本浏览框组件的重要	
11.4 超级列表框组件	300	属性	339
11.4.1 超级列表框组件的属性 ...	300	13.4.3 超文本浏览框组件的重要	
11.4.2 超级列表框组件的事件 ...	304	事件	340
11.4.3 超级列表框组件的方法 ...	304	13.4.4 超文本浏览框组件的重要	
第12章 数据库组件	310	方法	343
12.1 通用提供者组件与数据库		13.5 远程服务支持库	344
提供者组件	310	13.5.1 远程服务支持库概述	344

13.5.2 远程服务的方法	344	14.3.2 子程序参数的可空属性 ...	377
13.5.3 请求客户端的方法	349	14.3.3 子程序参数的参考属性 ...	380
13.6 网络传送支持库	352	14.3.4 子程序参数的数组属性 ...	381
13.6.1 网络传送支持库概述	352	14.4 子程序的指针应用	383
13.6.2 网络传送支持库中的 对象	352	第 15 章 易模块	385
13.6.3 “下载对象”的方法	353	15.1 易模块概述	385
13.6.4 “FTP 辅助对象”的方法 ...	357	15.2 易模块的调用方法	385
13.6.5 “FTP 上传对象”的方法 ...	359	15.2.1 易模块的安装	386
13.7 邮件接收支持库	360	15.2.2 易模块的调用	387
13.7.1 接收邮件的基本命令	360	15.3 编制自己的易模块	388
13.7.2 “邮件信息”数据类型的 方法	363	15.3.1 易模块的开发	388
13.8 局域网操作支持库	364	15.3.2 易模块的编译	389
13.9 BT 下载支持库	368	15.3.3 子程序改造为易模块	392
13.9.1 BT 下载支持库	368	第 16 章 API 函数调用	395
13.9.2 BT 下载支持库全局 命令	368	16.1 API 函数概述	395
13.9.3 “BT 下载”数据类型的提供的 方法	369	16.2 如何定义 API 函数	396
第 14 章 子程序调用	371	16.3 标准 DLL 库的应用	398
14.1 子程序的分类及建立	371	16.4 外部 DLL 库的应用	400
14.1.1 子程序的分类	371	第 17 章 易 DLL 编写及调用	402
14.1.2 子程序的建立	372	17.1 易 DLL、易模块、API 之间的 关系	402
14.2 子程序的调用	374	17.1.1 DLL 基本概念	402
14.2.1 子程序的调用	374	17.1.2 易 DLL 与易模块、API 间的 关系	402
14.2.2 子程序的递归调用	374	17.2 易 DLL 的开发与编译	402
14.3 子程序的参数	375	17.2.1 进入 DLL 编写环境	403
14.3.1 子程序的参数建立与 返回值	375	17.2.2 编写 DLL 源代码	403
		17.2.3 编译易 DLL	404
		17.3 如何调用易 DLL	405

第 1 章 易语言概述

易语言是一款全中文全可视化跨平台的编程工具,以中文作为程序代码表达的语言形式,由大连大有吴涛易语言软件开发有限公司设计开发,以“易”著称。它的特点是全中文化,入门要求低,几乎只要会使用计算机和文字输入的人都可以进行程序设计,而且它的开发语言也是全中文、生活化,读者在学习会深刻体会到它的“易”。

1.1 易语言概要

易语言早期版本的名字为 E 语言,最早版本的发布可追溯至 2000 年 9 月 11 日。可以说,创造易语言的初衷是进行用中文来编写程序的实践。从 2000 年至今,易语言已经发展到一定的规模:功能上、用户数量上都十分可观。易语言如此快速地发展,主要得益于它所具有的强大功能。

1. 全中文支持,全部自主知识产权

在计算机系统中,易语言所处的位置和地位同其他国外语言的编程工具是相同的,如图 1-1 所示。但易语言并不是把现存的编程工具进行表面汉化而成的。

和其他国外语言相比,“易语言”最大的不同是彻底中文化,且拥有自下而上的全部自主知识产权。也就是说,易语言的内核也是中文的,可以直接用中文写程序,如图 1-2 所示。使用者在编程的时候,不用了解英文和西方语法,也不用了解西方的思维模式,只需用汉语和中文思维方式便能写出软件。



图 1-1 全部自主知识产权

子程序名	返回值类型	公开	备注
按钮1_被单击			

变量名	类型	静态	数组	备注
总宽度	整数型			
总高度	整数型			

```
如果真 (是否已创建 (小窗口))
    图片框1.移动 (, 总宽度, 总高度)
    图片框1.图片 = 快照 (小窗口.取窗口句柄 (), 总宽度, 总高度)
    如果真 (通用对话框1.打开 ())
        写到文件 (通用对话框1.文件名, 图片框1.图片)
    标签1.标题 = “全中文支持,全部自主知识产权”
```

子程序名	返回值类型	公开	备注
------	-------	----	----

图 1-2 中文内核

2. 拥有自己的编译器

易语言拥有自己独立的高质量编译器,源程序被直接编译为目的机器的 CPU 指令,不存在任何速度瓶颈和安全隐患,如图 1-3 所示。

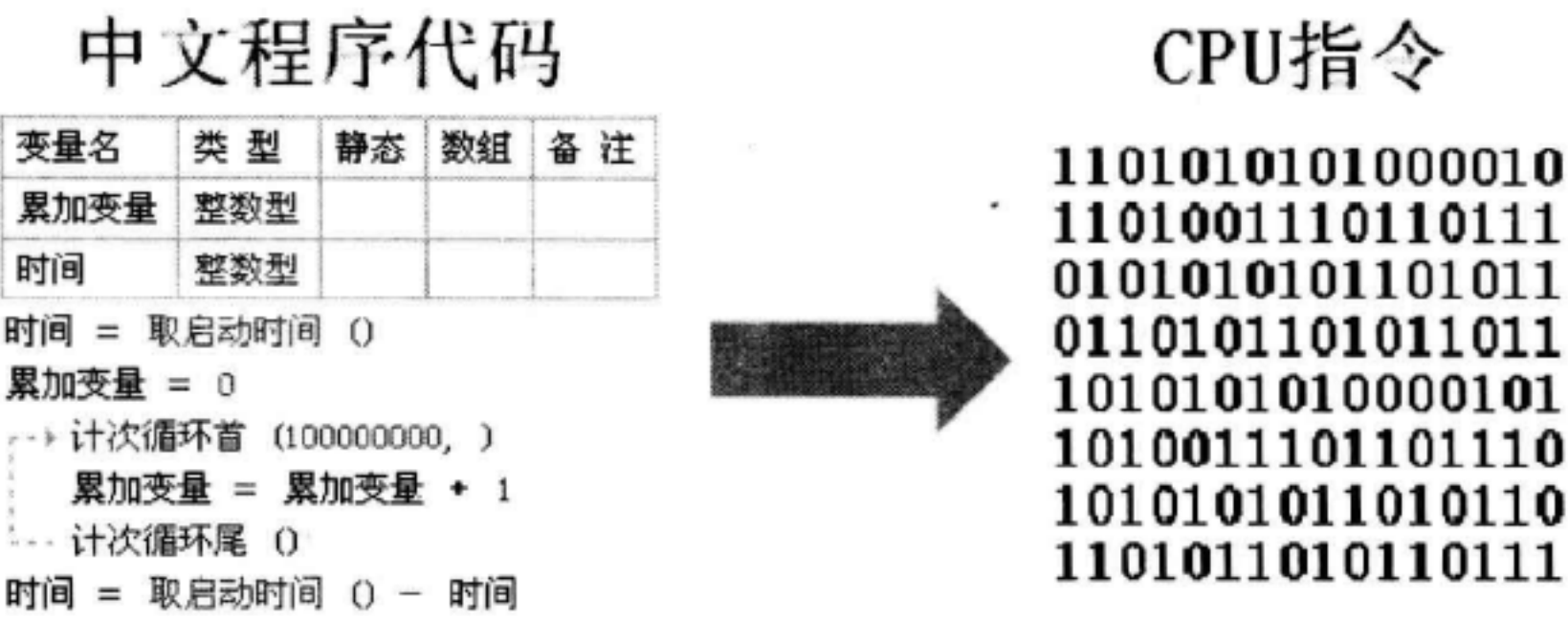


图 1-3 独立的编译器

众所周知,编译器是编程语言中最核心、最关键的部分,易语言拥有的独立编译器不仅效率高,而且可以编译出与操作系统平台无关的可执行代码,如图 1-4 所示。



图 1-4 高效编译器

3. 跨平台编程

易语言可跨平台编程,现已同时支持 Windows 和 Linux 程序开发,如图 1-5 所示,不再依赖特定的操作系统,为国家推广 Linux 操作系统提供了应用软件开发工具。

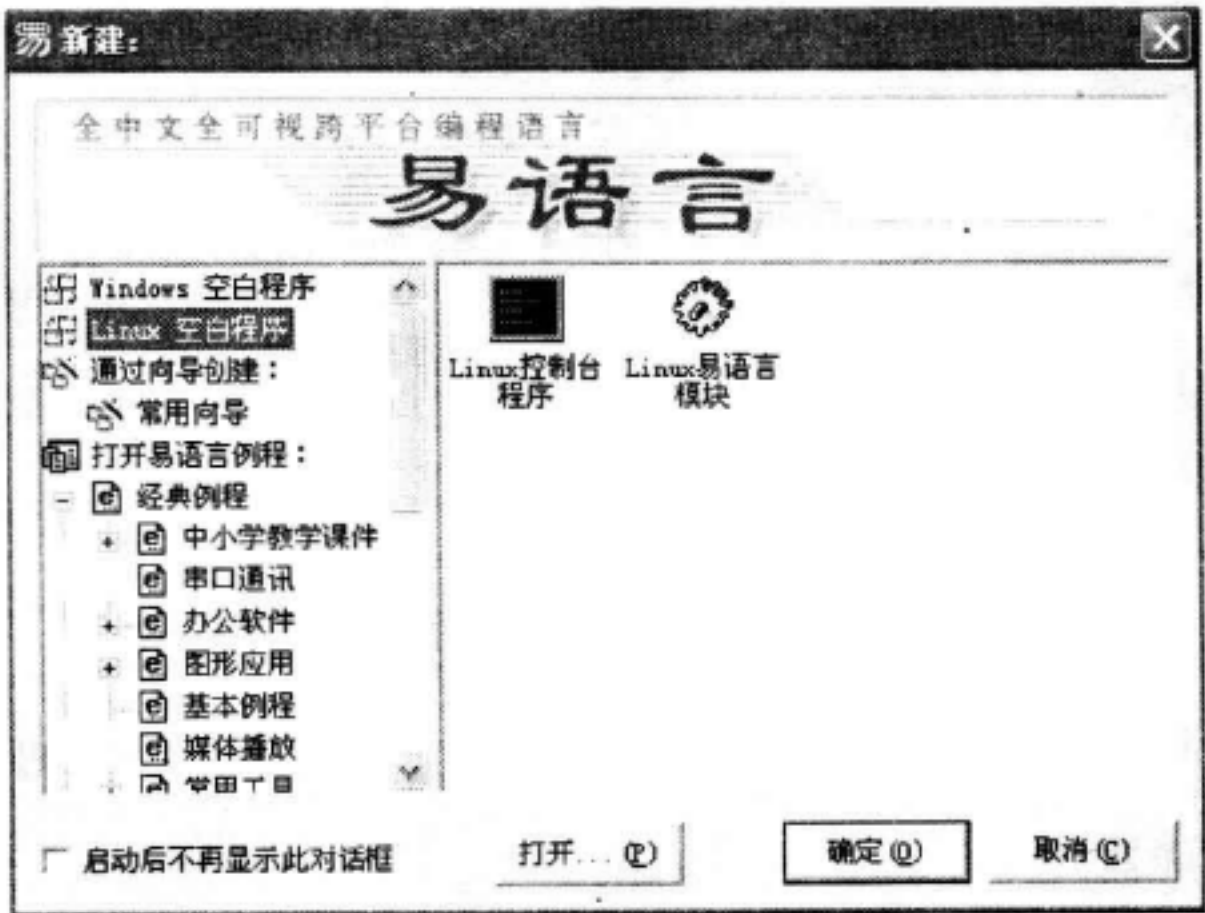


图 1-5 Linux 平台编程

4. 拥有自己的数据库系统,且支持访问现有所有数据库

数据库是保存、管理数据的最核心部件,易语言除了支持现今所有大型数据库,还具有从最底层自行开发的,具有完全自主知识产权和核心技术的数据库系统,现在该数据库性能已经能与曾经在中国大地上风靡一时的 Fox 系列数据库相媲美,并且还在继续不断的完善中,为我国发展属于自己的强大安全实用型数据库奠定基础。使用易语言数据库的校务管理系统软件如图 1-6 所示。



图 1-6 使用易语言数据库的校务管理系统

5. 内置专用输入法,支持中文语句快速录入

易语言已完全解决了输入速度的问题,中文语句的输入速度绝对不亚于英文语句的输入速度。易语言内置专用输入法,如图 1-7 所示,方便了用户快速输入程序,彻底解决了中文语句输入速度慢的问题。

易语言的程序编辑器具有智能纠错、规范格式、名称管理、自动跟踪、自动修改名称、自动弹出智能语法输入提示等功能。例如,图 1-8 中第一行的程序代码可以以图示中第二行的方式输入:

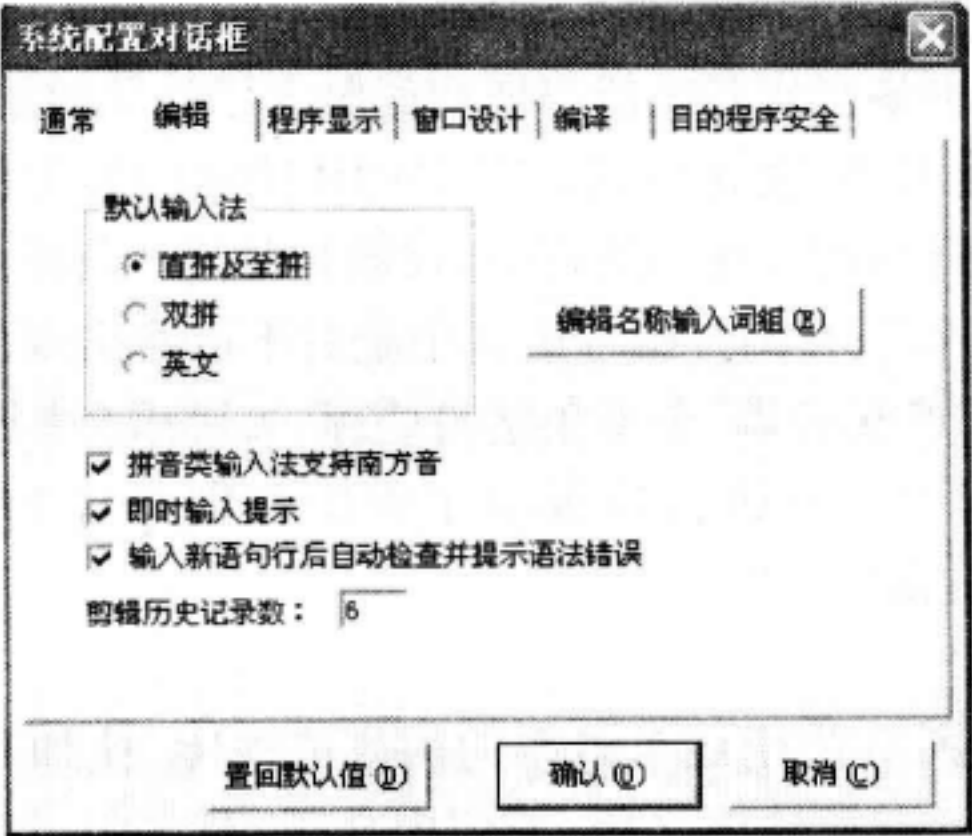


图 1-7 内置输入法

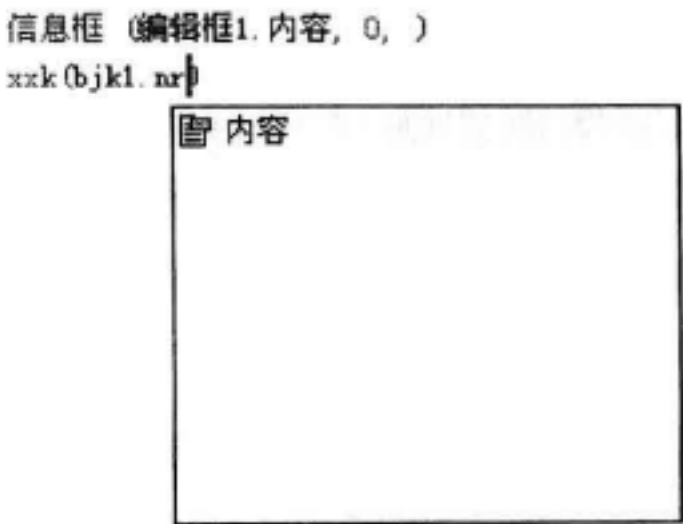


图 1-8 快速中文输入

系统内置的自动名称管理能够对用户所定义的各类名称进行跟踪管理。例如:假设程序中现存一个名为“取最大值”的子程序,而且在很多地方都调用了该子程序。现在用户根据需
要想把该子程序更改为另外一个名称,在传统的编程语言中,用户需要先更改该子程序的名称,然后搜索整个应用程序,逐一找到使用了该子程序的地方,把名称相应地改变过来。在易语言中,用户只需要更改子程序名称即可,程序中其他所有使用了该子程序的地方,其名称都将被自动快速地更改过来。

6. 全可视化编程

一般的可视化编程语言,仅支持图形用户界面的可视化设计操作。而易语言除了支持界面设计的可视化,还支持程序流程的即时可视化。易语言用户在编写程序的过程中,可以即时看到当前程序的运行流程及路线,以助于培养编程思路,提高解决编程问题的能力。对于学习编程语言的人来说,流程图是理顺程序设计思路、明确逻辑关系的最好办法。而易语言可以做到程序流程的“即输(输入)即画”,方便了用户。易语言程序的可视化编程界面如图 1-9 所示。

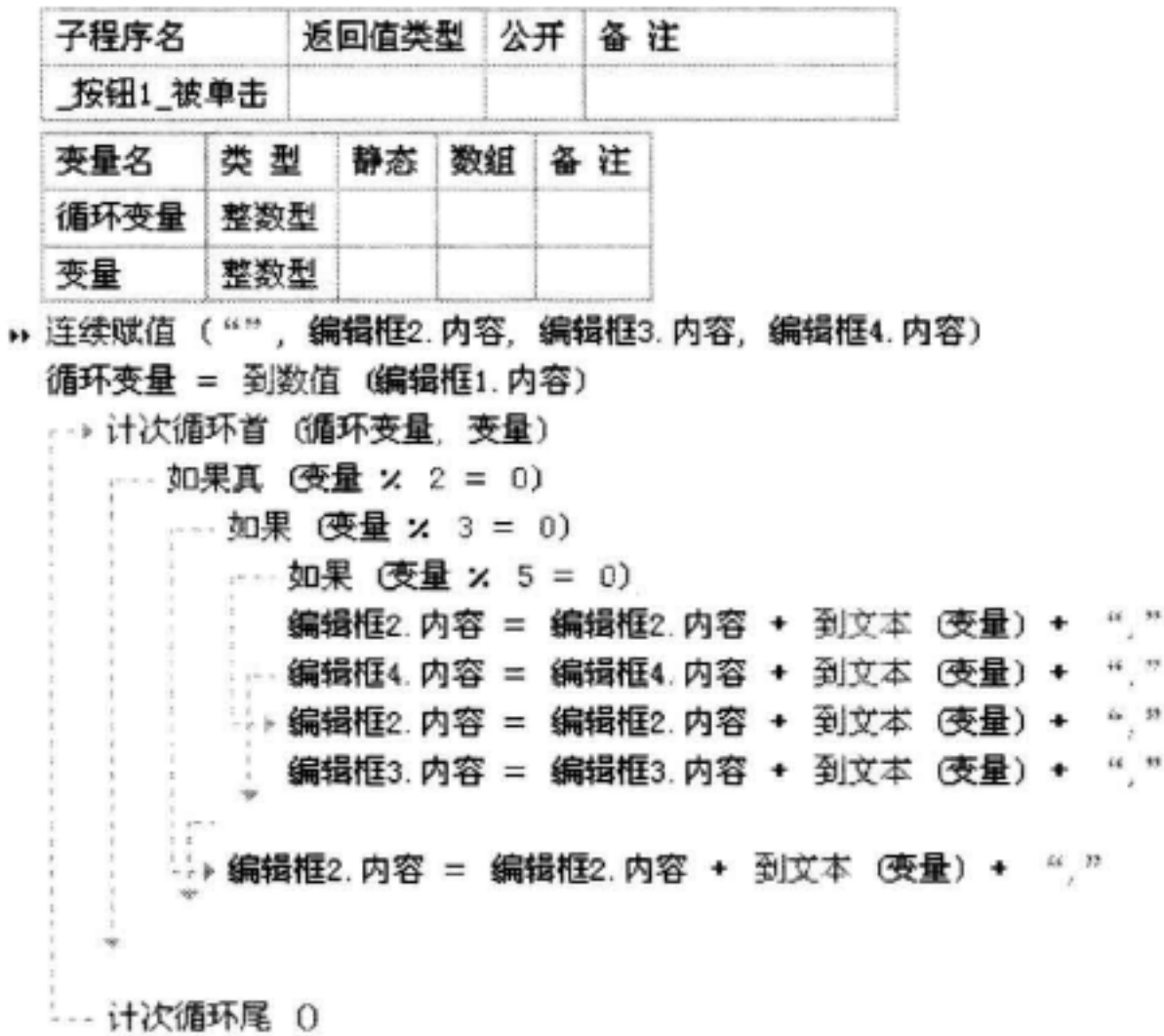


图 1-9 可视化编程界面

7. 中文本土化特色支持

作为一款全中文的编程语言,易语言将中华文化和民俗习惯彻底融入了计算机语言领域,用户不再需要被强制学习和掌握国外的文化习惯、表达方式。图 1-10 所示内容为易语言本土化特色命令的演示结果。从演示结果中可以看到,在易语言中,日期和时间全部采用中国人自己的“年/月/日”格式而非欧美的“月/日/年”;在国外编程语言中绝对不可能出现的汉语发音处理、汉字简繁处理、大陆与港澳台汉字字符集处理、全半角字符处理、人民币金额处理等等中国特色的功能已经被大量加入到易语言中。易语言还提供了农历日期组件和命令,如图 1-11 所示,能轻松实现显示和转换农历日期。

8. 多种语言支持

易语言并不局限于在中国国内发展,在所有使用远东语系的国家或民族,比如日本及韩国,易语言可以很方便地进行本地化。现在已有简体中文版、繁体中文版、英文版、日文版,分别用作支持相应地域的用户开发软件。图 1-12 所示为易语言日文版的设计界面。

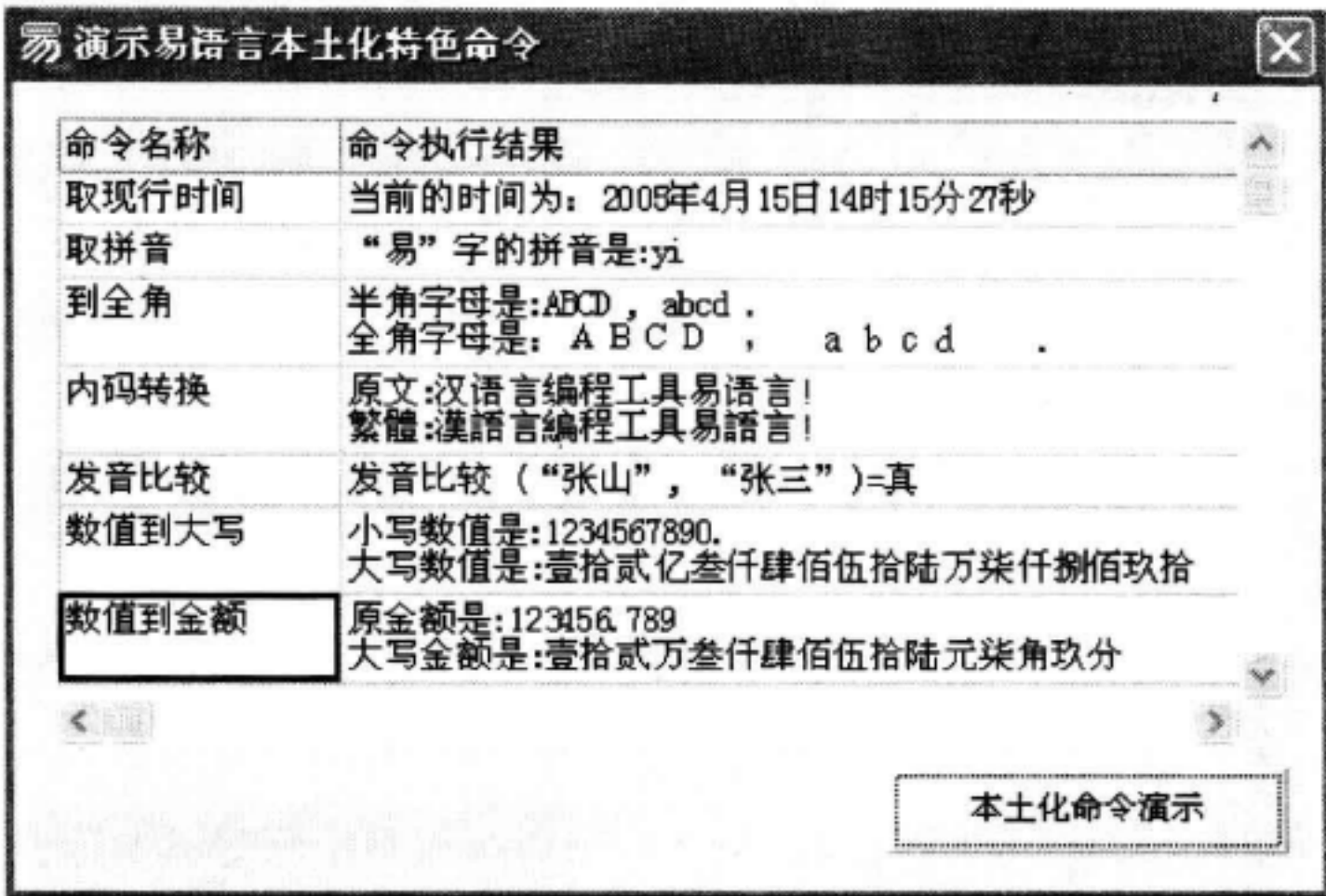


图 1-10 中文本土化特色



图 1-11 农历日期组件

9. 与其他编程语言相互融合、相互补充

易语言与其他各种编程语言不是对立的关系,而是一种相互融合、相互补充的关系,体现在编程理念和技术层次上的双重互通。

(1) 编程理念上的互通:易语言与其他国外编程语言一样,支持当今先进的编程理念,譬如面向对象的程序编写方法、面向事件的消息处理机制等,了解、学习易语言对掌握其他编程语言具有桥梁的作用。

(2) 技术层次上的互通:在易语言中,可以调用由其他编程语言编写的程序,同样,其他编程语言也可以调用由易语言编写的程序。由此而来,用户可以同时使用易语言和其他编程语言来共同完成同一项开发工作。

图 1-13 所示为通过调用易语言的内部码转换功能来实现汉字内码转换的 VB 程序,该 VB 程序运行后的结果如图 1-14 所示。

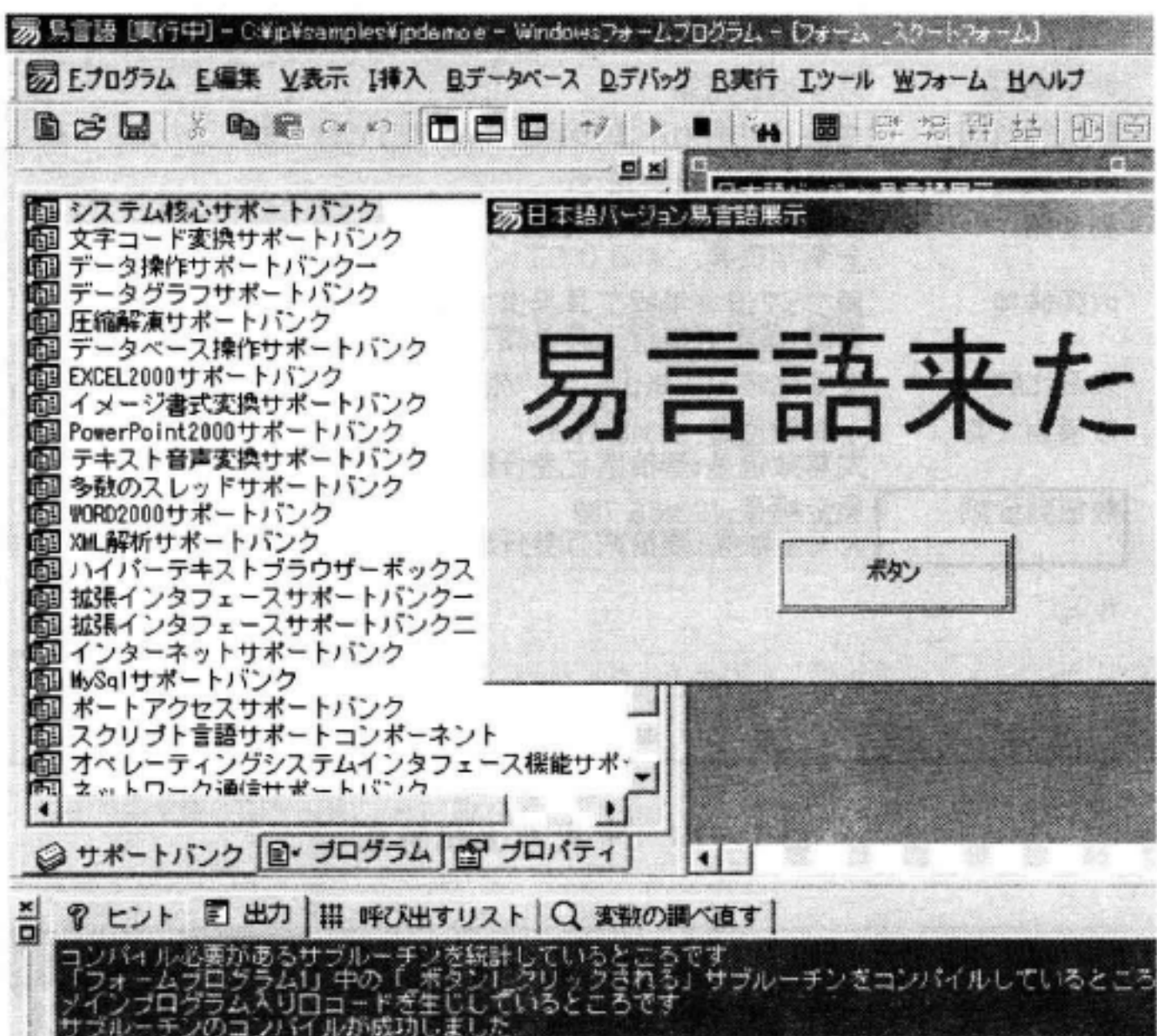


图 1-12 日文版易语言

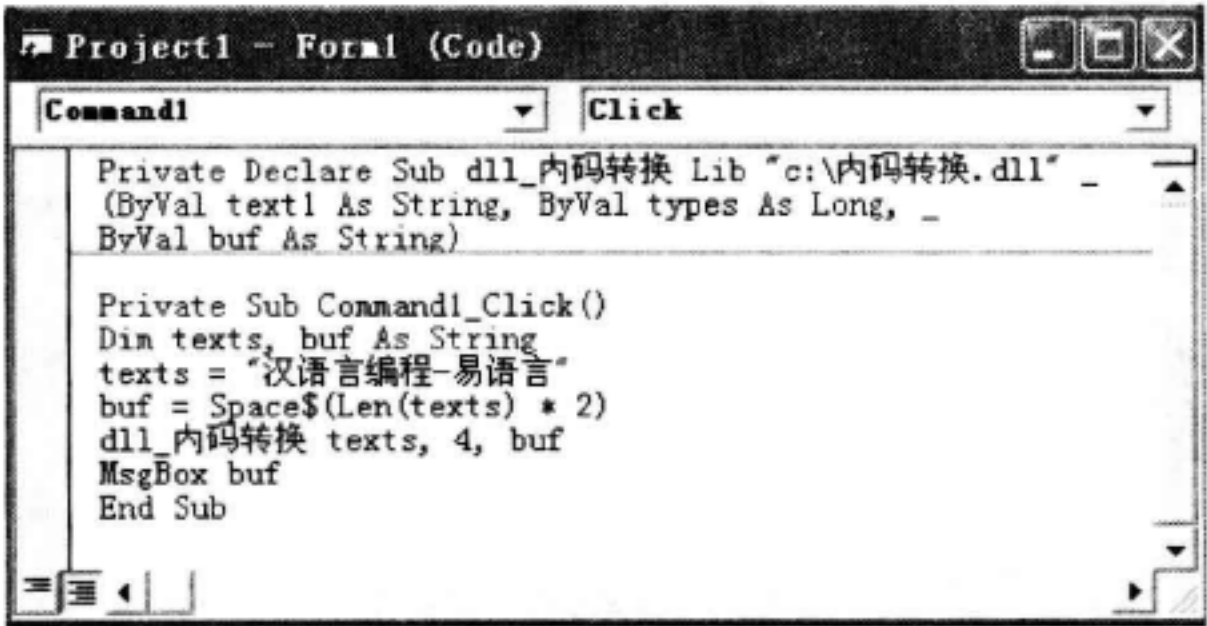


图 1-13 实现汉字内码转换的 VB 程序

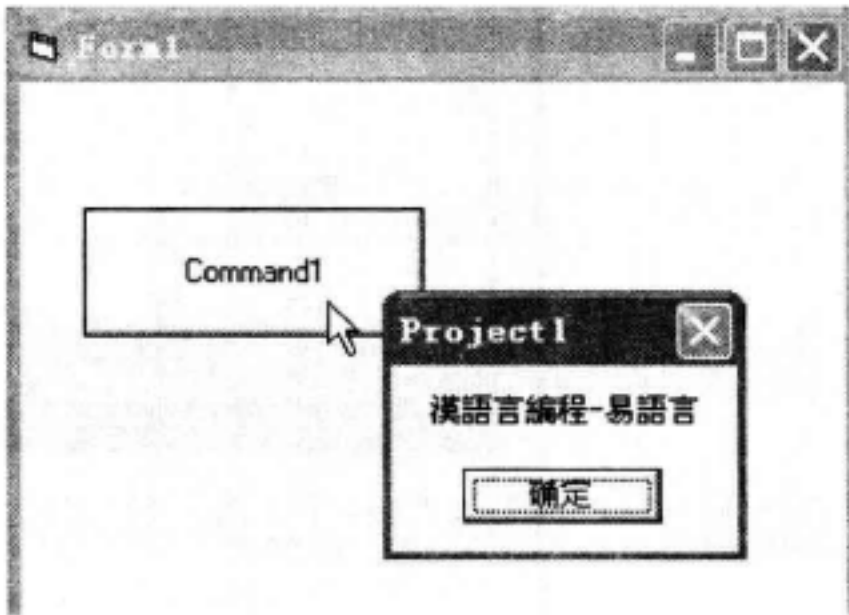


图 1-14 VB 程序的运行结果

同样在易语言中也可以调用其他编程语言程序,图 1-15 所示为调用一段 VC 程序:

```
////////////////////////////////////  
// The one and only CFromucApp object  
  
CFromucApp theApp;  
  
void WINAPI FromUC ()  
{  
    AfxMessageBox ("嗨! 我是 Visual C++ 程序!");  
}
```

图 1-15 易语言调用 VC 程序

10. 充分利用现有资源,增强易语言实力

在计算机系统中,已经存在很多现有功能组件和功能接口,譬如微软公司用于扩展其 VB 编程语言功能的 OCX 组件,用作充分利用 Windows 操作系统性能的应用程序功能接口

(API),用作与其他外部对象和应用程序交流功能的 COM 协议等。

易语言从不封闭或排外,它能够充分利用所有这些现有资源,从而站在了巨人的肩膀上。通过利用这些不计其数的现有资源,提升了易语言的实用功能。

OCX 组件本来是微软公司用于扩展 VB 语言功能的,一个组件即实现了一类功能或一种界面,现在易语言也同样全面支持,所有的 OCX 组件放到易语言里面来同样能够被正常使用。这样所带来的好处显而易见,现今世界上已存在的成千上万个 OCX 组件都能立即被易语言所用,从而极大地扩充了易语言的功能。图 1-16 所示为在易语言组件面板中加入的一些 OCX 组件(框中)。

11. 支持调用 API 底层函数

API(Application Programming Interface,应用程序编程接口)是一些预先定义的函数,目的是提供应用程序与开发人员基于某软件或硬件的以访问一组例程的能力,而又无需访问源码,或理解内部工作机制的细节。API 是操作系统的基础,易语言支持调用 API 可直接利用操作系统的绝大部分功能(图 1-17)。



图 1-16 易语言中加入的 OCX 组件



图 1-17 易语言 API 取所有特定目录

12. 支持微软 COM 协议

COM 协议是微软公司定义的用作对象、应用程序之间交互功能的标准协议,通过支持这个协议,易语言就能够访问现今世界上无数支持该协议的功能组件和应用程序。

图 1-18 所示内容为利用 COM 对象调用 Excel 例程的易语言程序,调用成功后的结果如图 1-19 所示。

13. 模块化开发支持大型软件项目的分工协作

现在大型软件项目的实施一般是分工协作开发,为了支持这一点,易语言提供了模块化开发支持。易语言中的模块称为易模块。通过使用易模块,用户可以将常用的代码封装起来重

窗口程序集名	保留	保留	备注
启动窗口程序集			
变量名	类型	数组	备注
Excel对象	对象		
工作簿对象	对象		
工作簿对象	对象		
表对象	对象		
选择区域对象	对象		
行列对象	对象		
工作表对象	对象		

子程序名	返回值类型	公开	备注
打开_被选择			

如果真 (Excel对象.是否为空 () = 真)

Excel对象.创建 ("excel.application",)

Excel对象.写属性 ("visible", 真)

如果真 (通用对话框1.打开 () = 真)

工作簿对象 = Excel对象.读对象型属性 ("Workbooks",)

工作簿对象 = 工作簿对象.对象型方法 ("Open", 通用对话框1.文件名)

子程序名	返回值类型	公开	备注
退出_被选择			

Excel对象.方法 ("Quit",)

结束 ()

图 1-18 利用 COM 对象调用 Excel 例程



图 1-19 运行结果

复使用到其他程序,或提供给第三方使用,或用作开发大型软件项目中的某个部分,然后在软件项目的封装阶段将所有这些模块组织编译成为一个完整程序(图 1-20)。

14. 代码即文档源程序风格统一

由于支持全可视化编程,并且系统强制转换显示程序流程,统一语法格式,因此在易语言

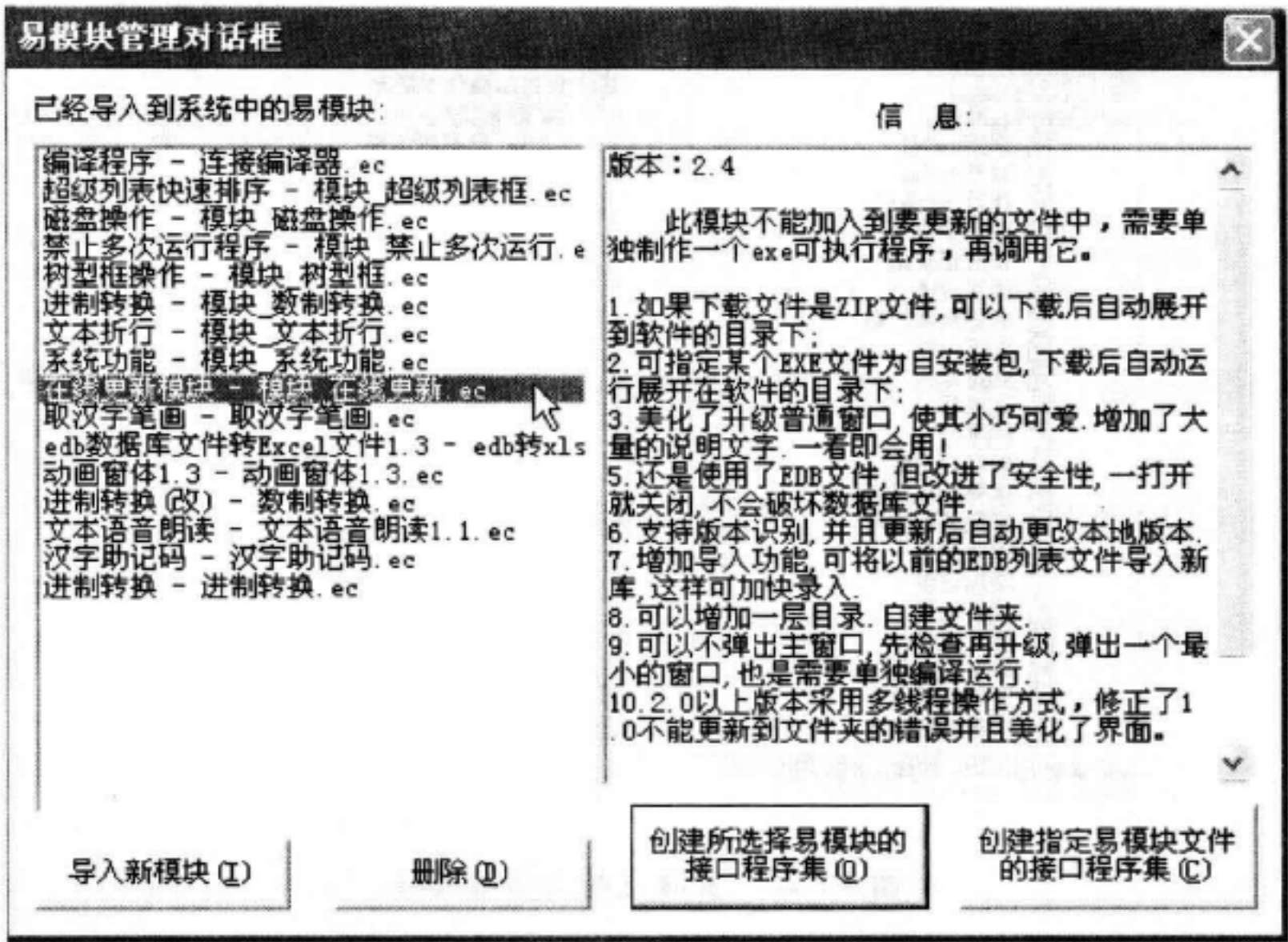


图 1-20 建立并导入到系统中的易模块

里面任何人编写的任何源程序的样式和风格都完全一致,便于交流与协作。这是其他任何编程语言所不具备而易语言独有的特性。

无论您写代码时的格式多么不一致,大小写不分、空格乱空、句点不标准,在您按下回车键后,立即预编译处理,转换为标准样式和风格,便于交流与协作,从而保证了任何人所编写的易语言程序格式都完全一致,效果如图 1-21 所示。

15. 强大的数据库功能支持

数据库是开发数据库应用程序的基石,易语言不仅自身携带有独立的数据库,而且还支持现今所有的大型数据库系统。图 1-22 所示为支持数据库功能的内建组件。

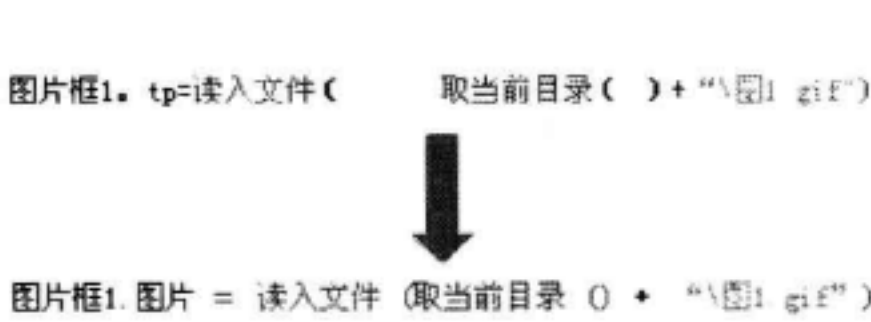


图 1-21 风格统一的代码

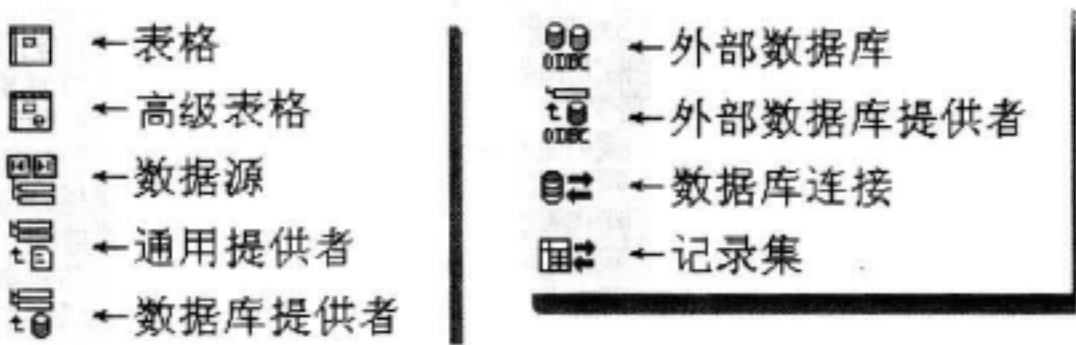


图 1-22 支持数据库功能的组件

除此之外,易语言内部对著名开放源代码 MySQL 数据库以及其他各种大型数据库都是直接支持的,如图 1-23 所示。

16. 完善的网络、端口通信和互联网功能支持

具有完善的网络、端口通信和互联网功能支持,为开发工业控制、局域网、互联网应用程序提供保证。图 1-24 所示为易语言中丰富网络组件与支持库。

图 1-25 演示了用“超文本浏览框支持库”写的多页浏览器。图 1-26 演示的是易语言的串口通信与并口通信功能在工业控制方面的应用。图 1-27 演示了用“服务器”和“客户”组件写的易语言即时通信工具。图 1-28 所示为利用易语言的 BT 下载支持库写的 BT 下载软件。



图 1-23 直接支持大型数据库



图 1-24 直接支持大型数据库

17. 多媒体功能支持强大

易语言的基本组件中包含了图像显示,动画显示的功能,支持大多数的图片格式,如JPG格式,GIF格式,BMP格式,易语言新增加的支持库中也有图像格式转换命令,可轻松转换图像格式达十几种。易语言的基本支持库提供音乐播放命令,简单的命令即可实现为程序配上声音。易语言的窗口动画、GIF动画支持可为用户编写多媒体类软件提供方便。

图 1-29 和 1-30 分别为使用 DirectX 支持库写的打飞机游戏和超级玛丽游戏。

文本语言支持库包括语音识别和文本语音朗读,语音识别可以将语音转化为计算机能够



图 1-25 易语言编写的多页浏览器

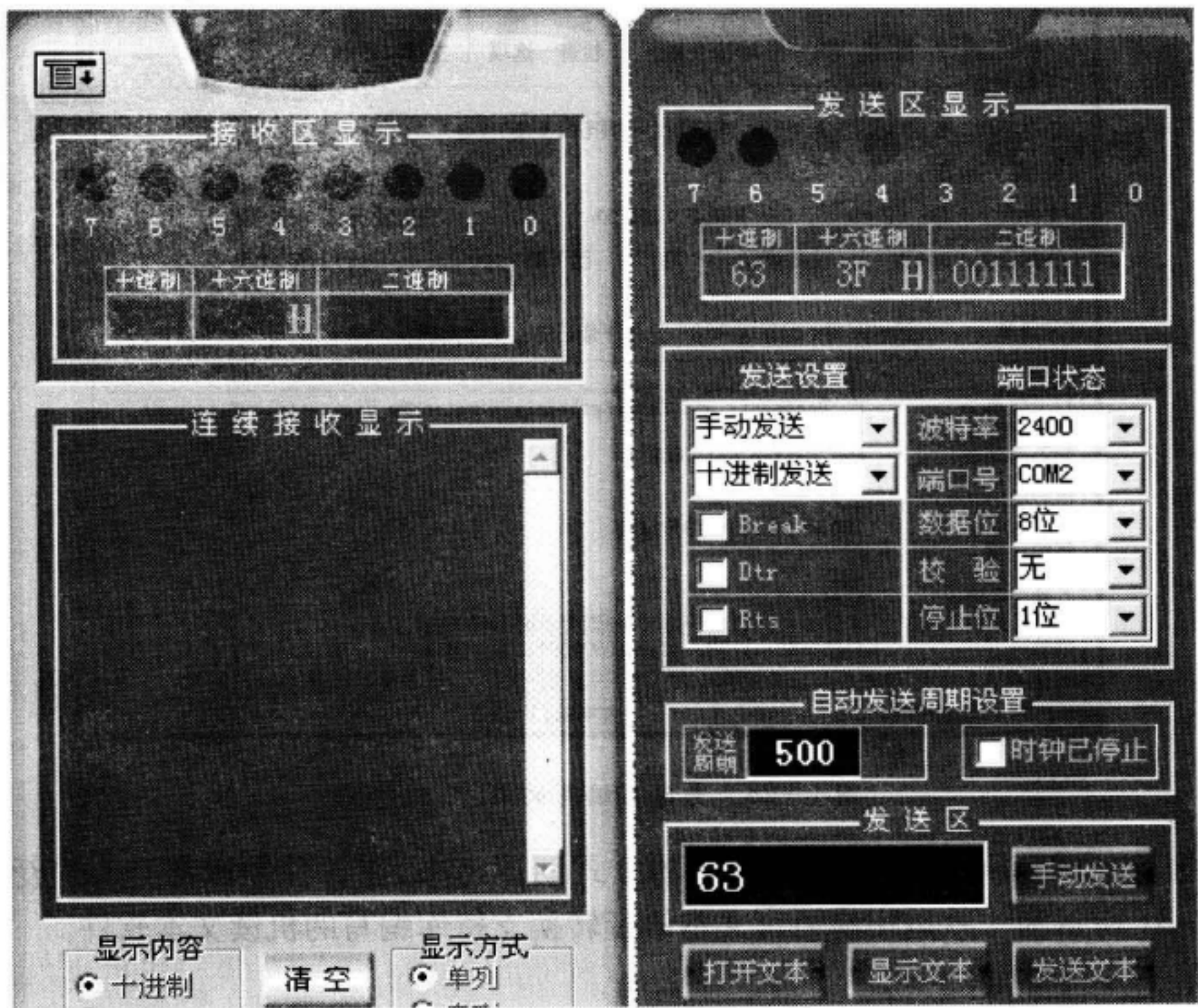


图 1-26 易语言在工业控制中的应用

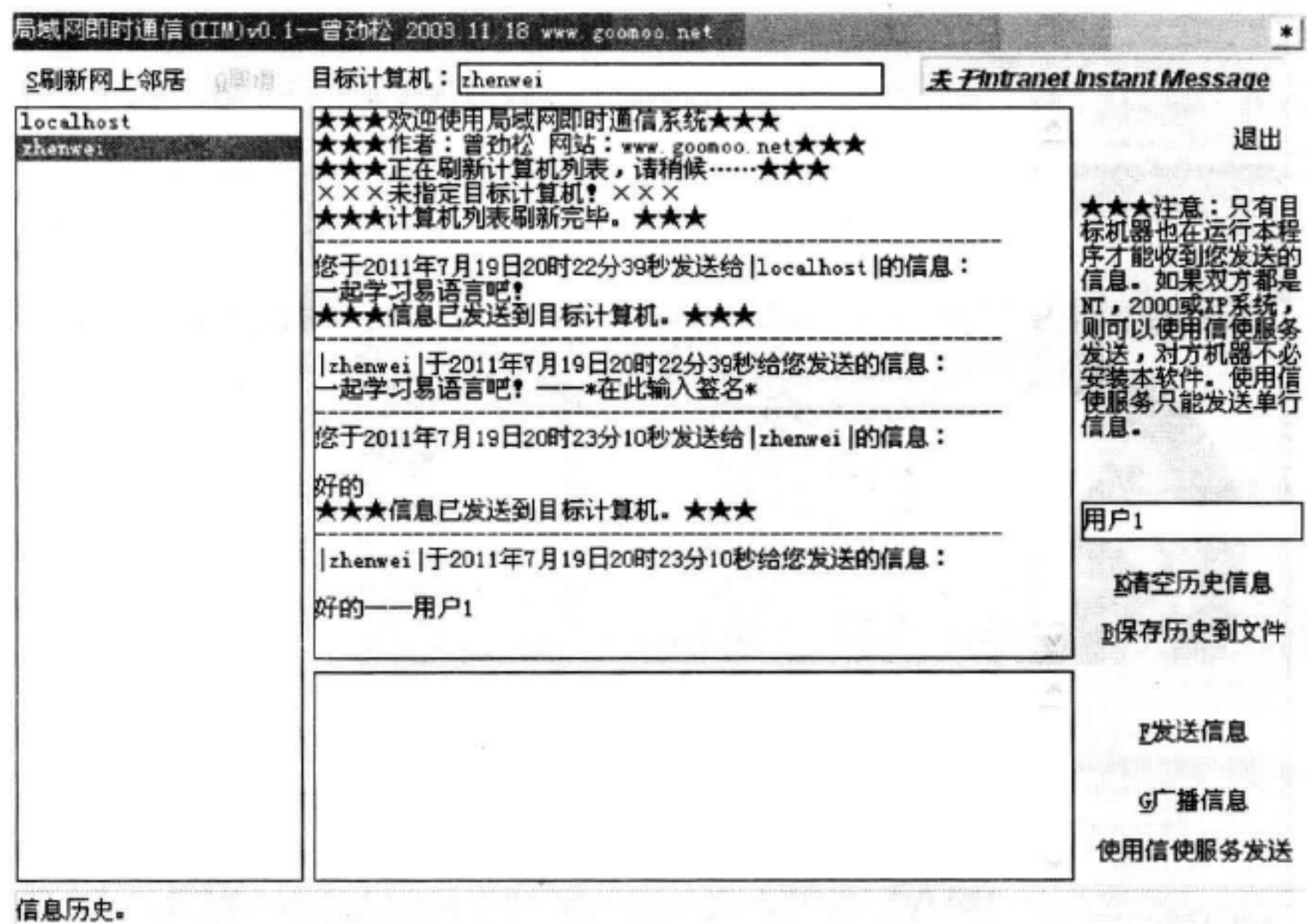


图 1-27 易语言即时通信工具



图 1-28 易语言编写的 BT 下载软件

识别的数据,语音朗读可以将文本以语音的方式朗读出来,在各种领域都具有广泛的应用前景。图 1-31 所示为一款运用易语言文本语音转换支持库编写的机读文本软件。

18. 第三方支持库

易语言的具体功能涉及到方方面面,现已具有近千条命令和大量的程序组件,为用户开发应用软件提供极强大的支持! 此外易语言具有专用类似 OCX 的组件格式,称为支持库,易语言的功能可以通过增加支持库的方式无限扩充,易语言的功能也可以通过调整支持库的方式



图 1-29 打飞机游戏

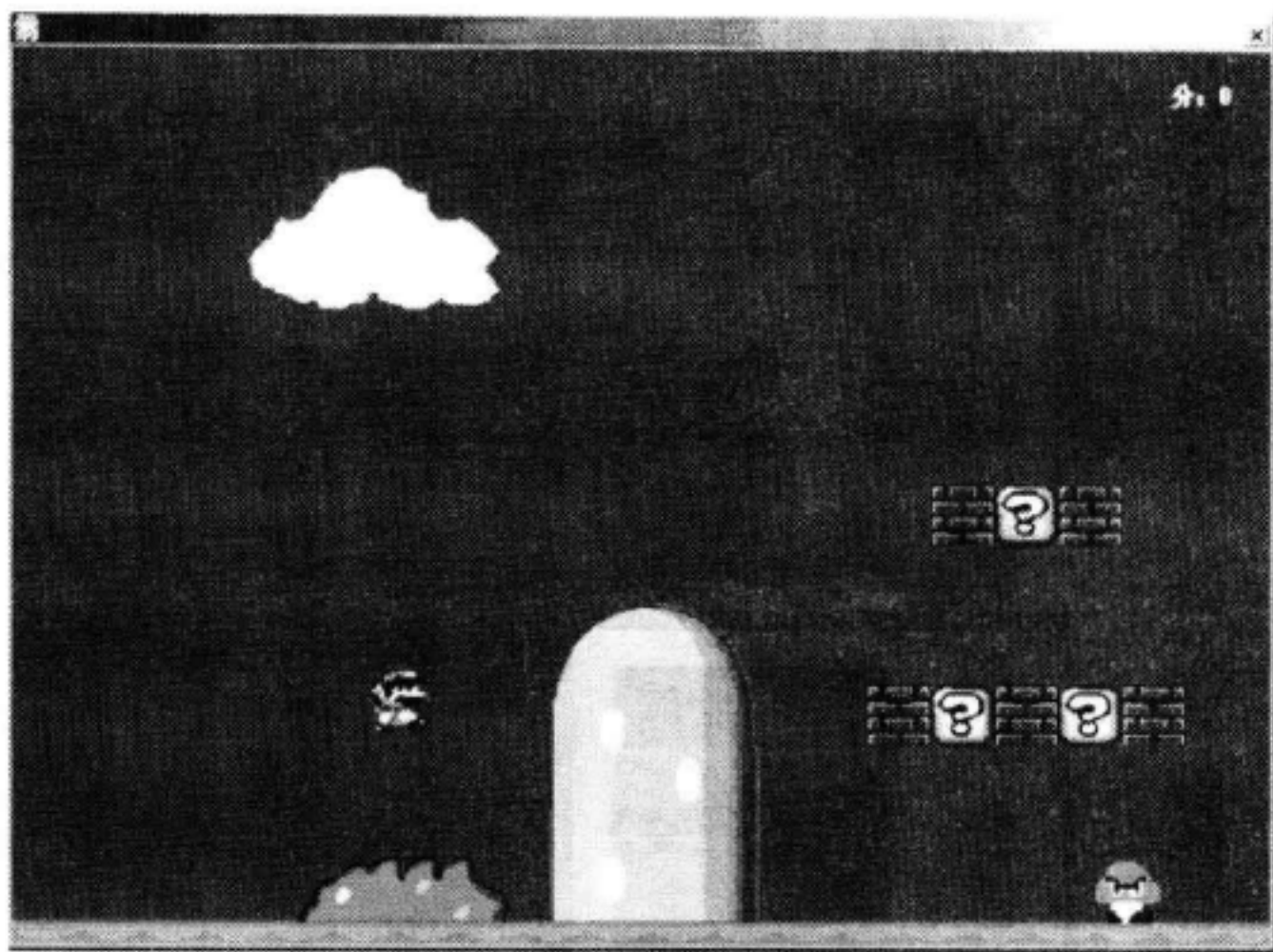


图 1-30 超级玛丽游戏

来适应各种行业和专业领域,轻松搭建易语言与行业应用软件对接的软件开发专业语言平台。易语言支持库的接口文档已经公布,任何第三方均可加入到壮大易语言的行列中来。图 1-32 演示了使用第三方支持库制作的图形菜单。图 1-33 为汽球提示框支持库的演示效果。

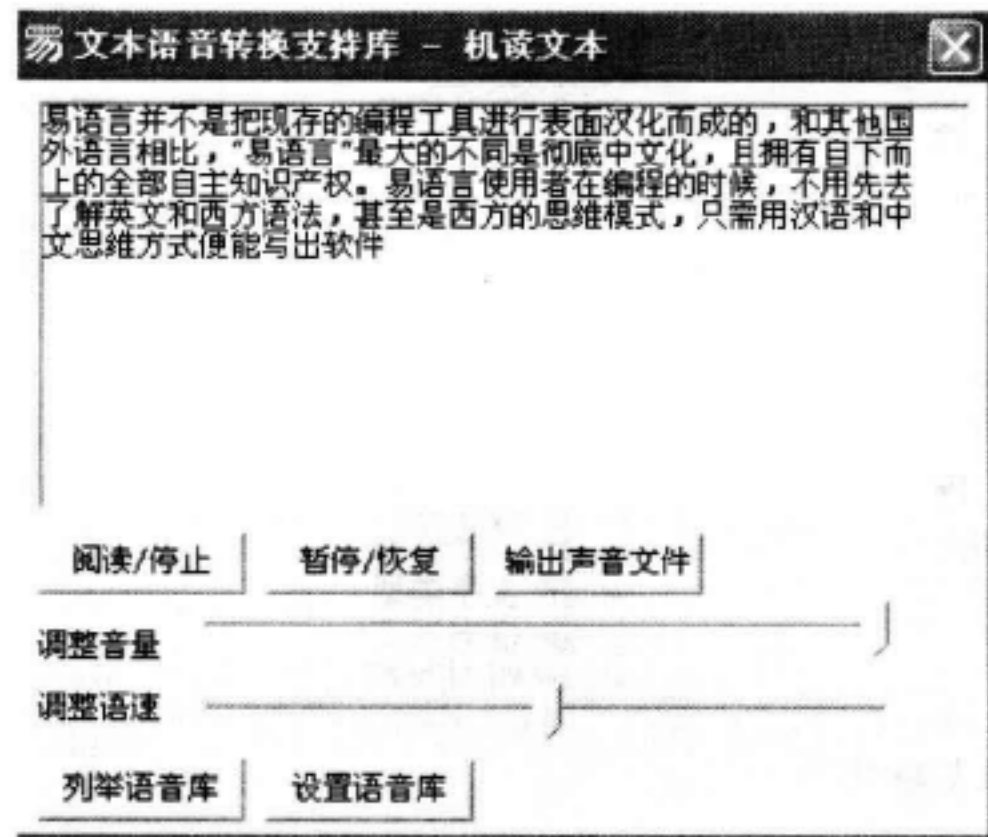


图 1-31 文本语音转换支持库的应用

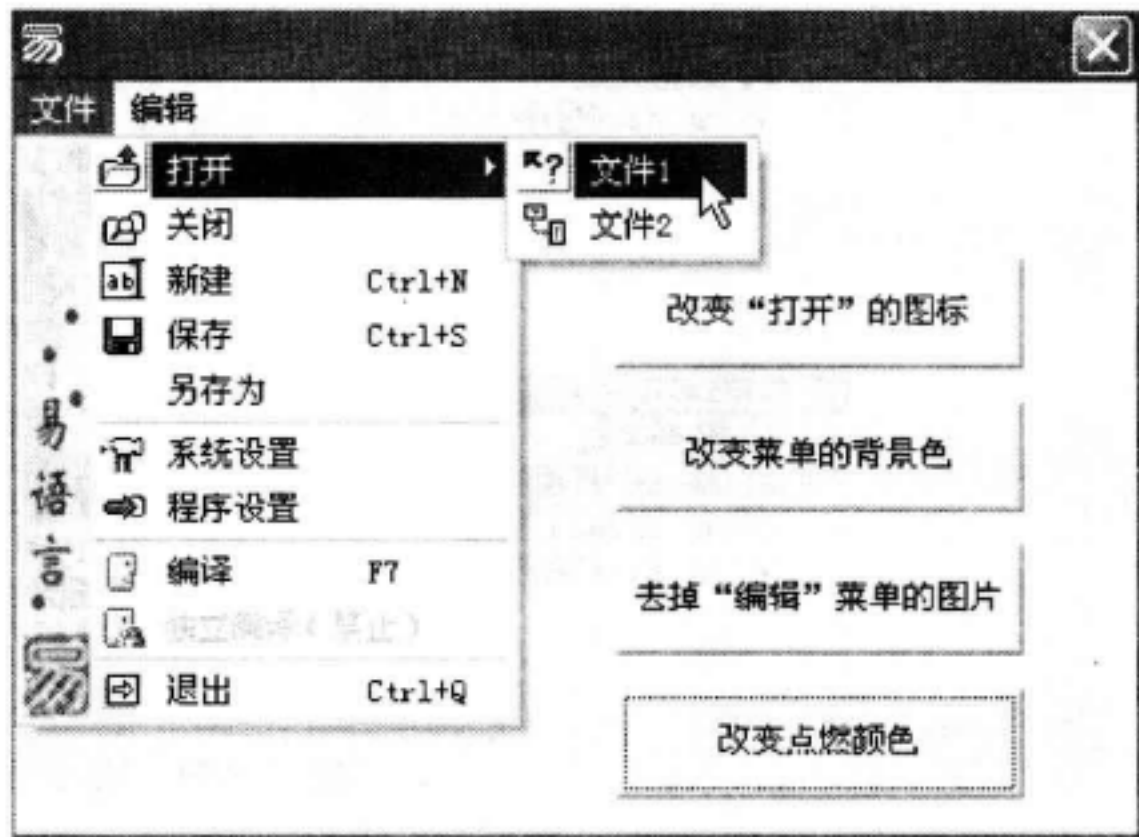


图 1-32 使用第三方支持库制作的图形菜单

19. 办公应用

易语言提供了一系列用作支持办公应用开发的功能,例如对 Office 系统中 Word, Excel, PowerPoint 应用软件等都有专门的支持库,如图 1-34 所示,使易语言用户可以直接用中文写代码来操纵这些软件。易语言还自行提供了一个字处理办公组件,使用它可以轻松开发办公类软件。图 1-35 演示了通过易程序操作 WORD 后的效果。

20. 数据处理

易语言支持对数据的常见处理,包括压缩解压、完整性校验、数字签名、数据加解密、XML 数据解析、正则表达式处理、数据库结构处理、无限位数的数值计算功能等。

(1) 压缩解压。易语言支持以标准 ZIP 格式对数据压缩与解压,如图 1-36 所示。使用

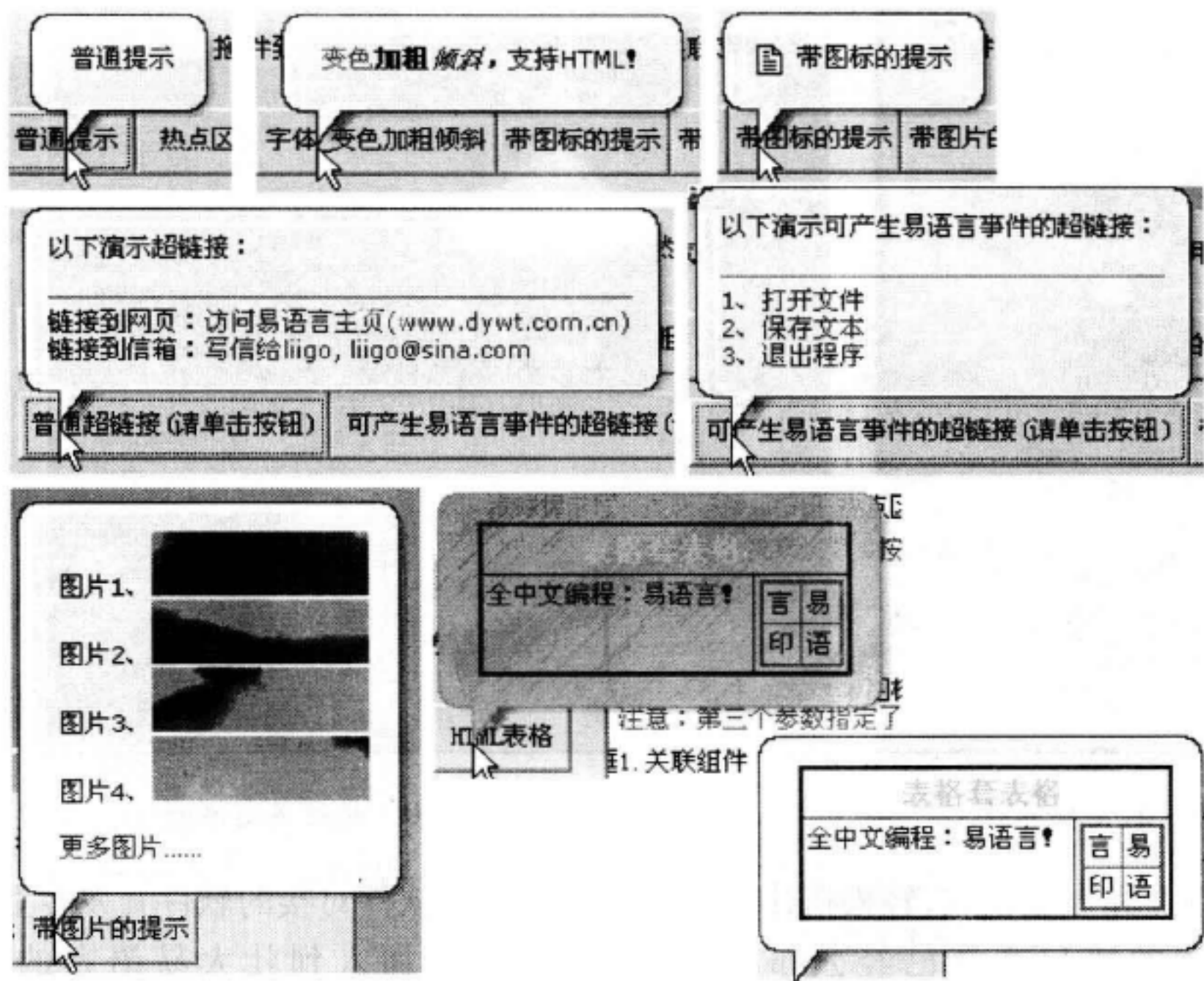


图 1-33 汽球提示框支持库的演示效果



图 1-34 办公应用支持库

压缩功能有利于传送资料分发到客户手中,也有利于使网络传输数据资料更小更快捷。使用标准 ZIP 格式可以保证压缩出来的文件能够被 WinRar 或 WinZip 等常见工具软件直接解压。

(2) 完整性校验。为了保证重要数据不被非法篡改,易语言支持标准 MD5 数据摘要算法和 RSA 非对称密钥算法,用户可以对其重要数据进行完整性校验和数字签名,如图 1-37、图 1-38 所示。

(3) 数据加解密。易语言支持标准的 DES 和 RC4 数据加密算法,可以被用户用作保证其重要数据的安全,如图 1-39 所示。

(4) 数据结构。为方便分析需要处理的对象的特征和对象间的关系,易语言提供数据结构支持库,如图 1-40 所示。

(5) 正则表达式。易语言支持使用正则表达式的语法检索文本,如图 1-41 所示。同时也能提供对 W3C 标准 XML 文件格式的操作支持,如图 1-42 所示。

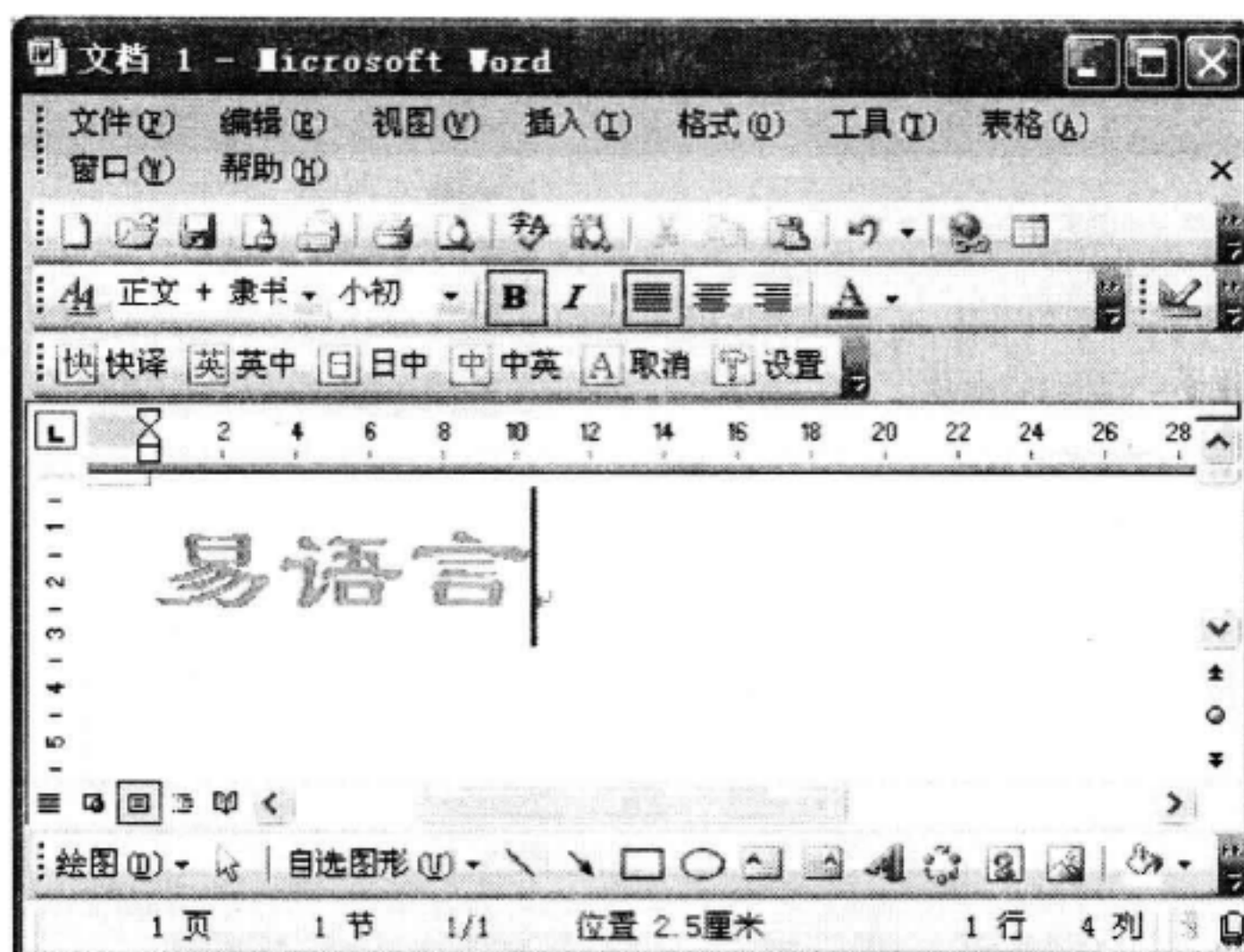


图 1-35 通过易程序操作 WORD

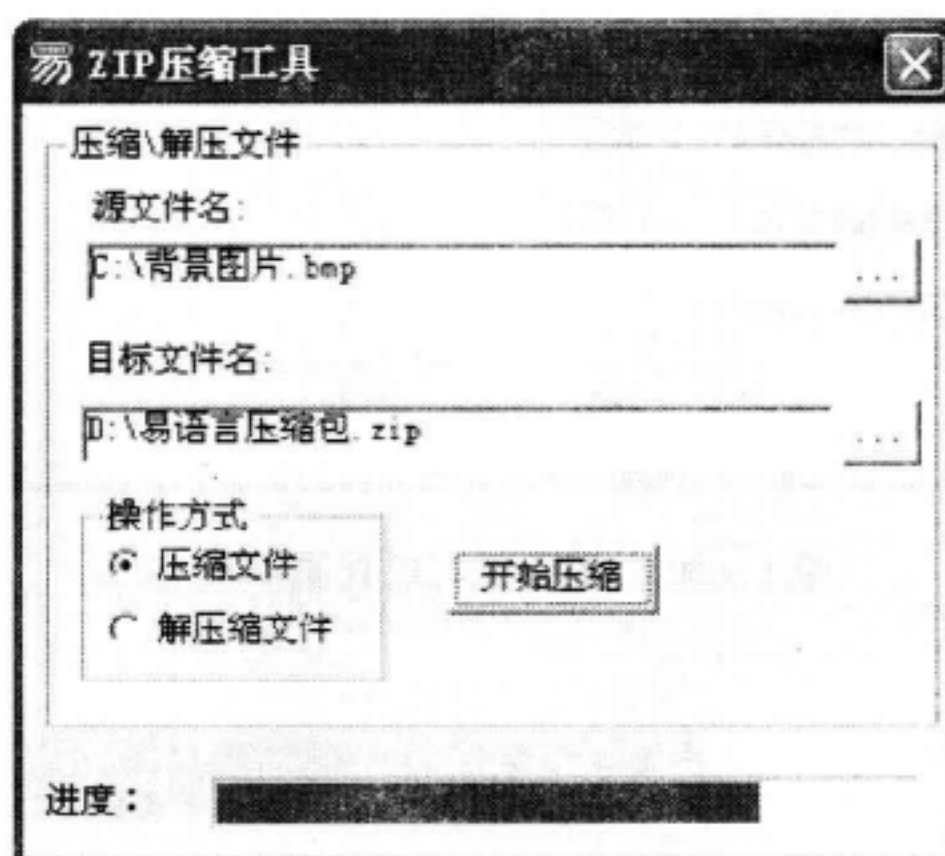


图 1-36 易语言实现数据压缩与解压



图 1-37 易语言实现 RSA 数字签名



图 1-38 易语言实现 RSA 签名认证

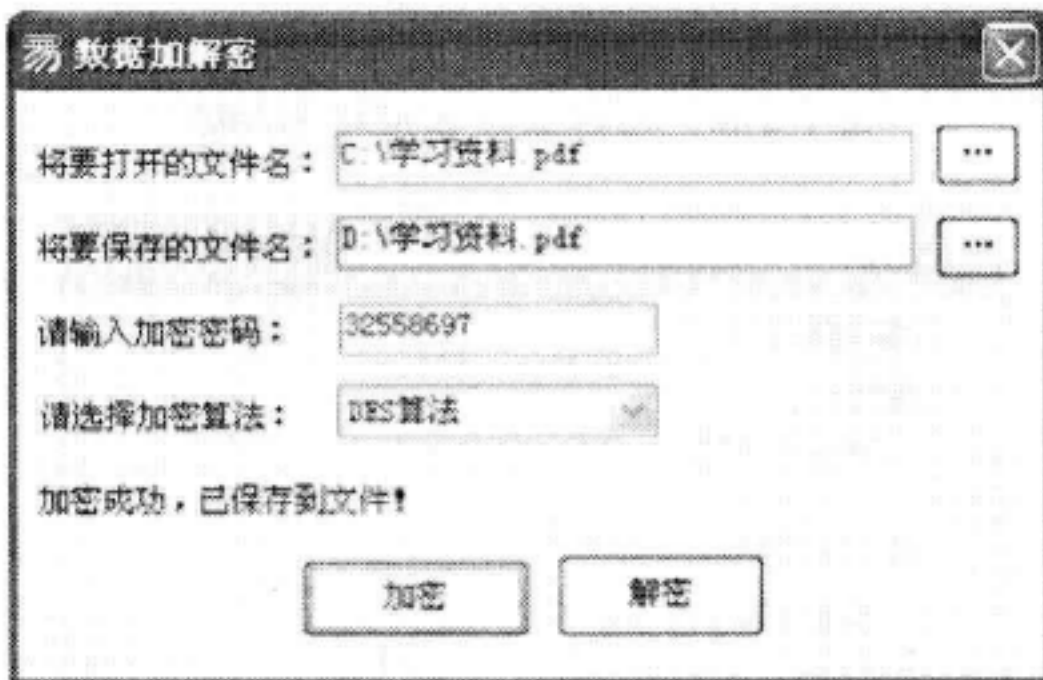


图 1-39 易语言实现数据加解密

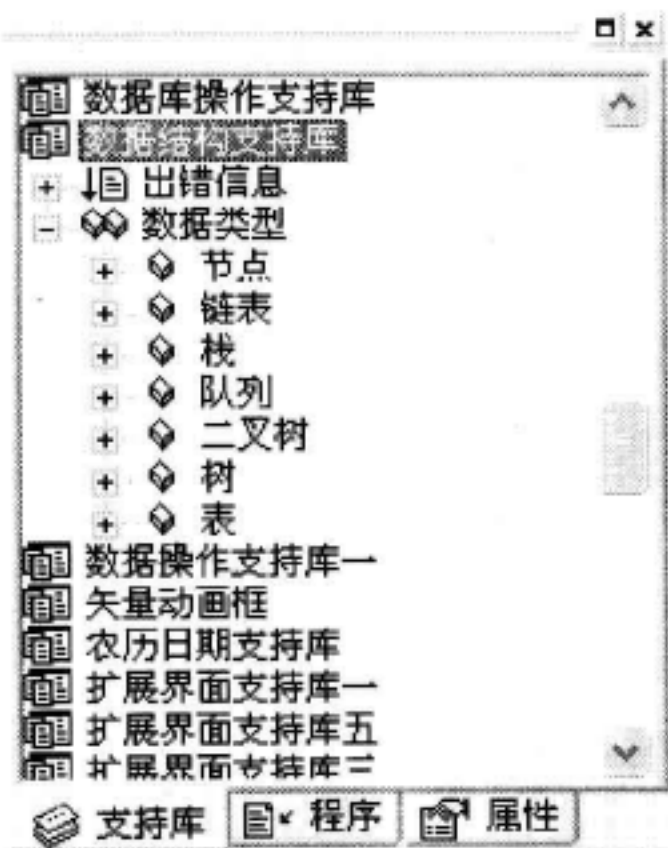


图 1-40 数据结构支持库

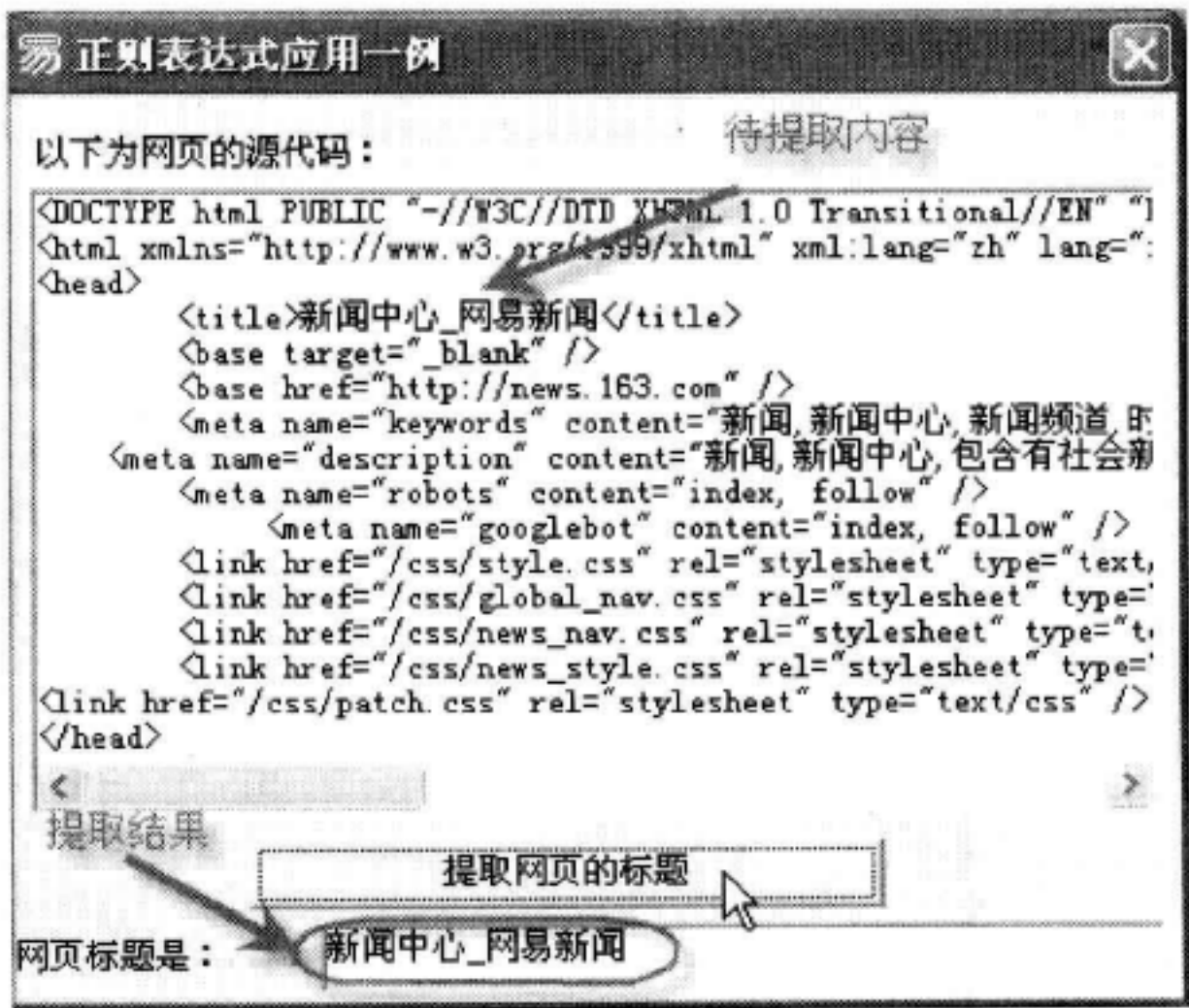


图 1-41 易语言实现文本检索



图 1-42 易语言对 W3C 标准 XML 文件格式的操作支持

1.2 易语言的下载与安装

步骤 1: 打开易语言汉语编程官方网站(<http://www.eyuyan.com>), 如图 1-43 所示。



图 1-43 易语言汉语编程官方网站主页

步骤2:单击“产品下载”选项,进入如图 1-44 所示的页面。



图 1-44 产品下载页面

步骤3:选择要下载的易语言版本,这里以最新的易语言 5.11 正式版为例,单击“易语言 5.11 完全版下载(包括知识库、多媒体教程),如图 1-45 所示,弹出如图 1-46 所示的对话框。

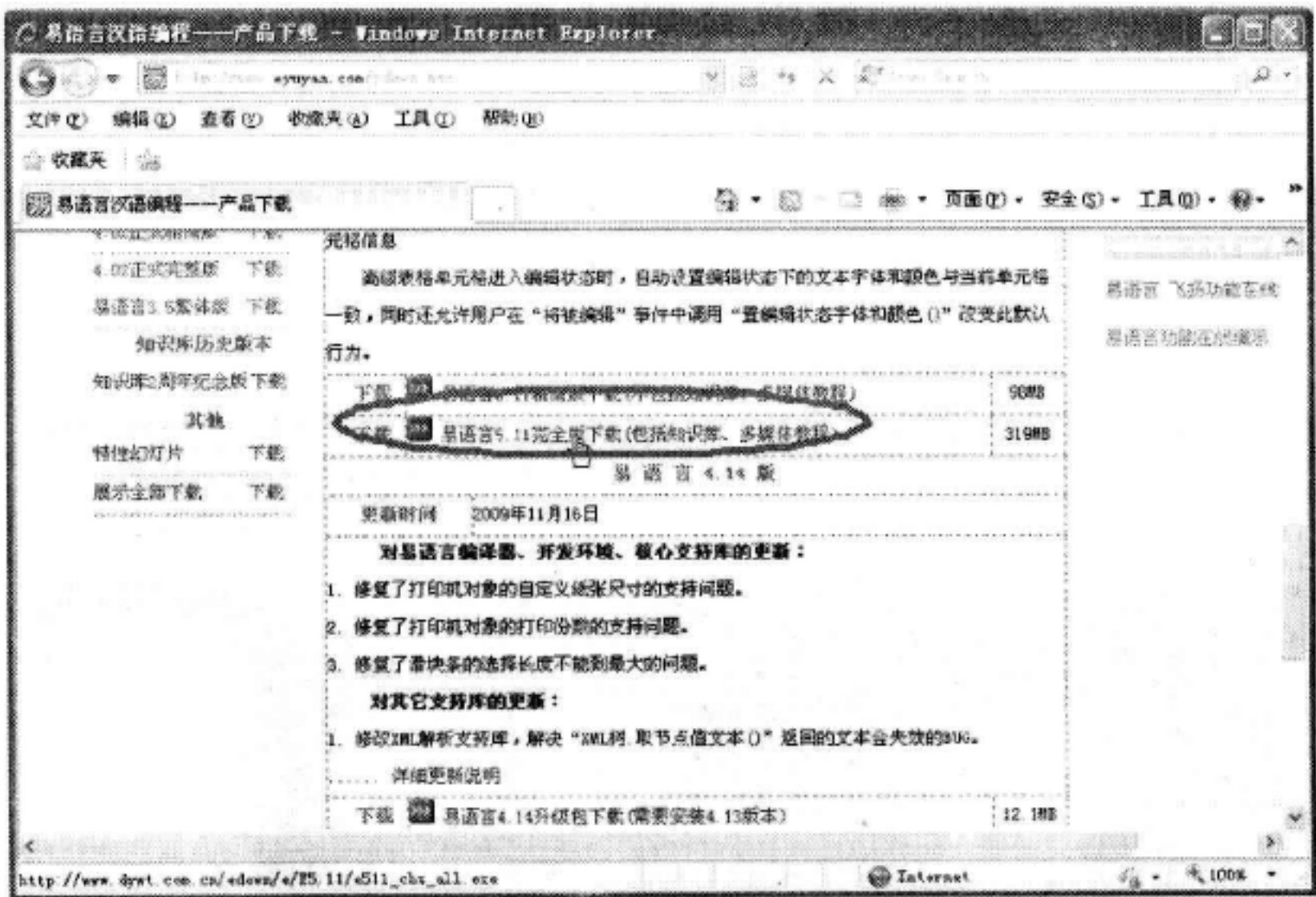


图 1-45 选择下载的易语言版本

步骤4:单击“保存”按钮,将“e511_chs_all.exe”另存为“易语言 5.11 完全版.exe”下载,下载后的安装文件如图 1-47 所示。执行目录下的“.exe”。

步骤5:弹出安装窗口,如图 1-48 所示,然后单击“下一步”按钮。

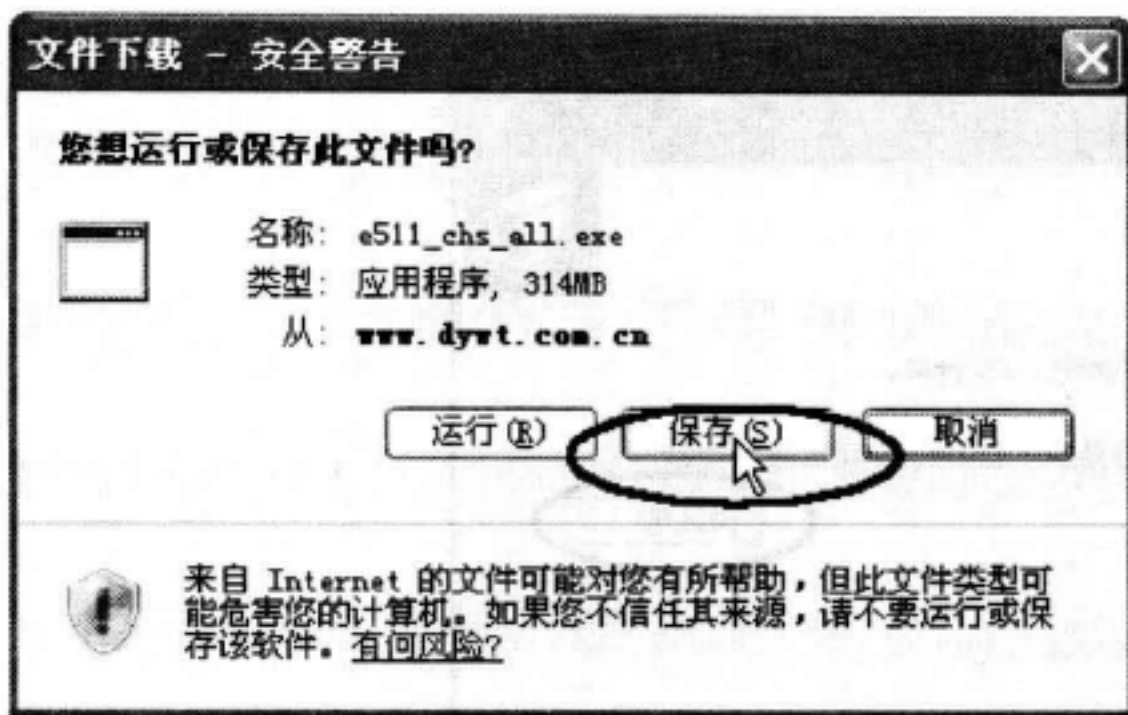


图 1-46 文件下载对话框

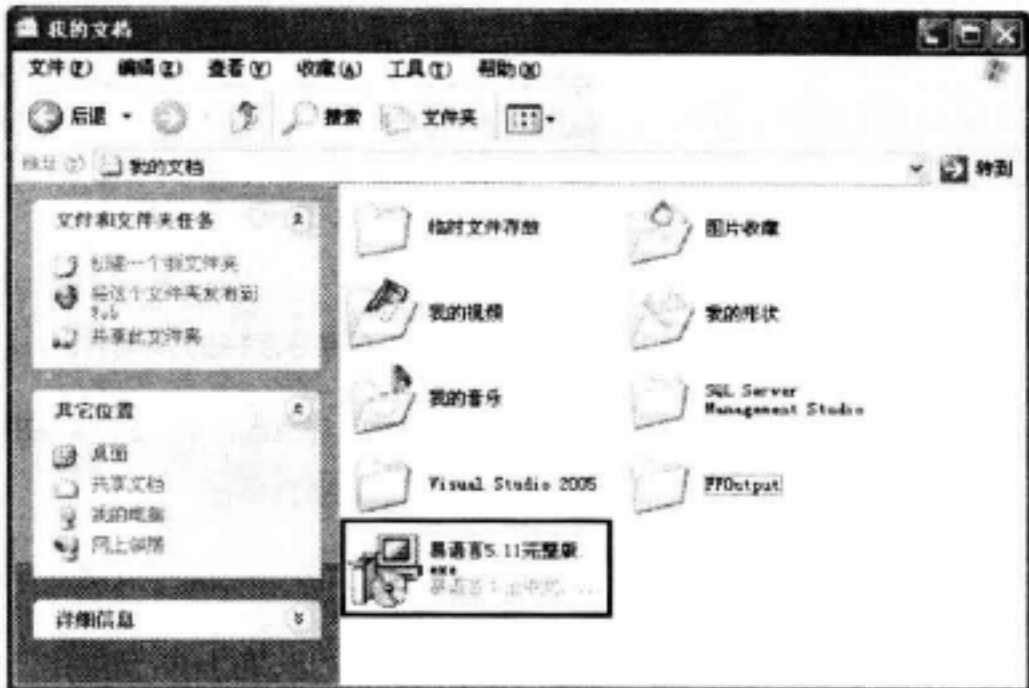


图 1-47 安装文件

步骤 6:弹出许可协议窗口,如图 1-49 所示,选择“我接受”单选框。

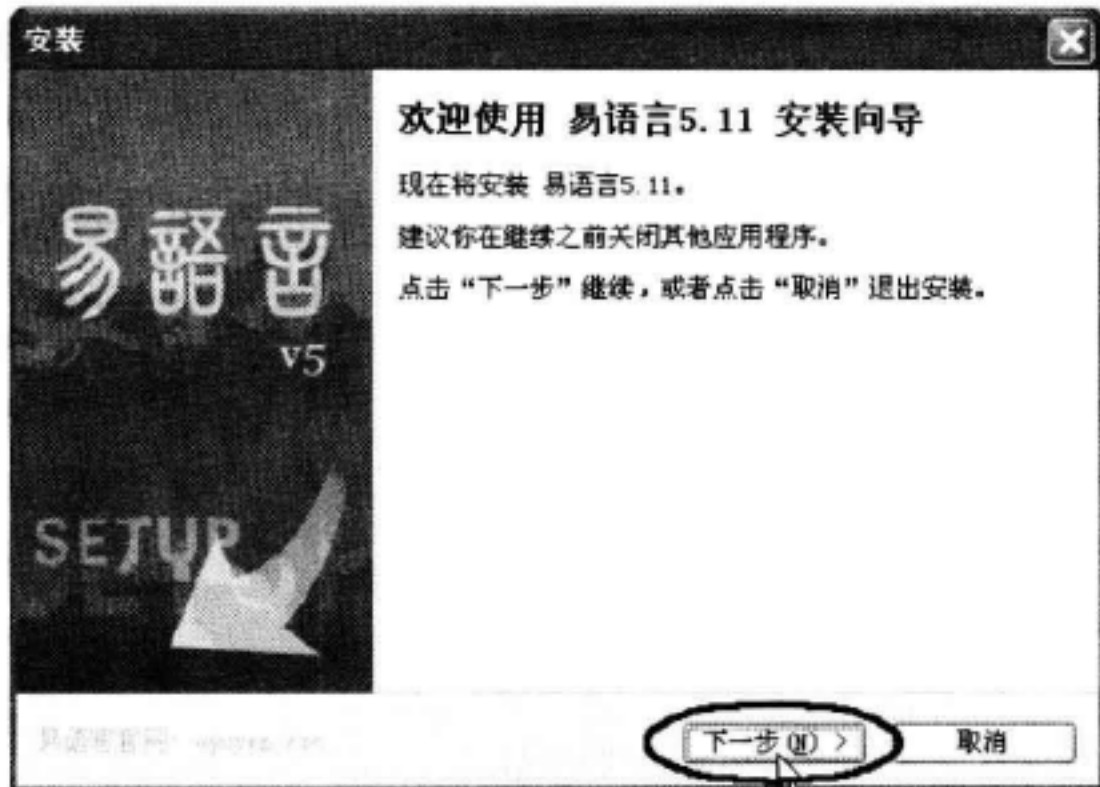


图 1-48 安装窗口

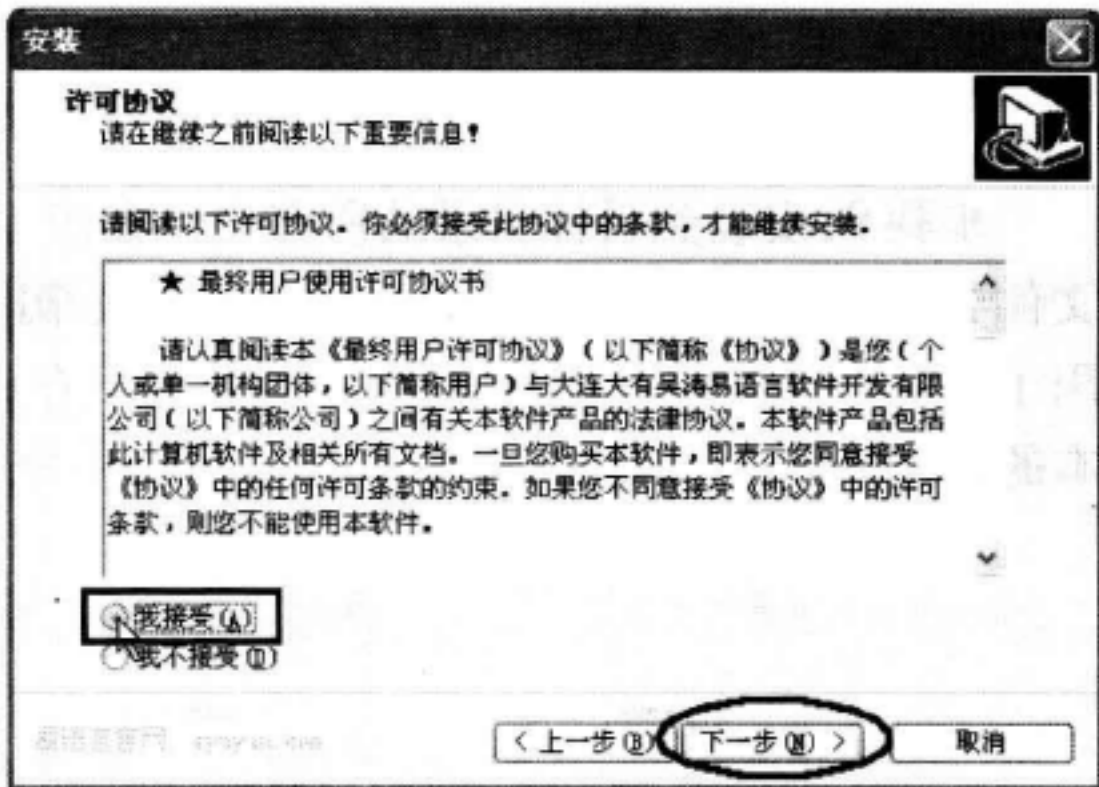


图 1-49 许可协议窗口

步骤 7:重要信息里介绍易语言新版本的新增功能,如图 1-50 所示,单击“下一步”按钮。

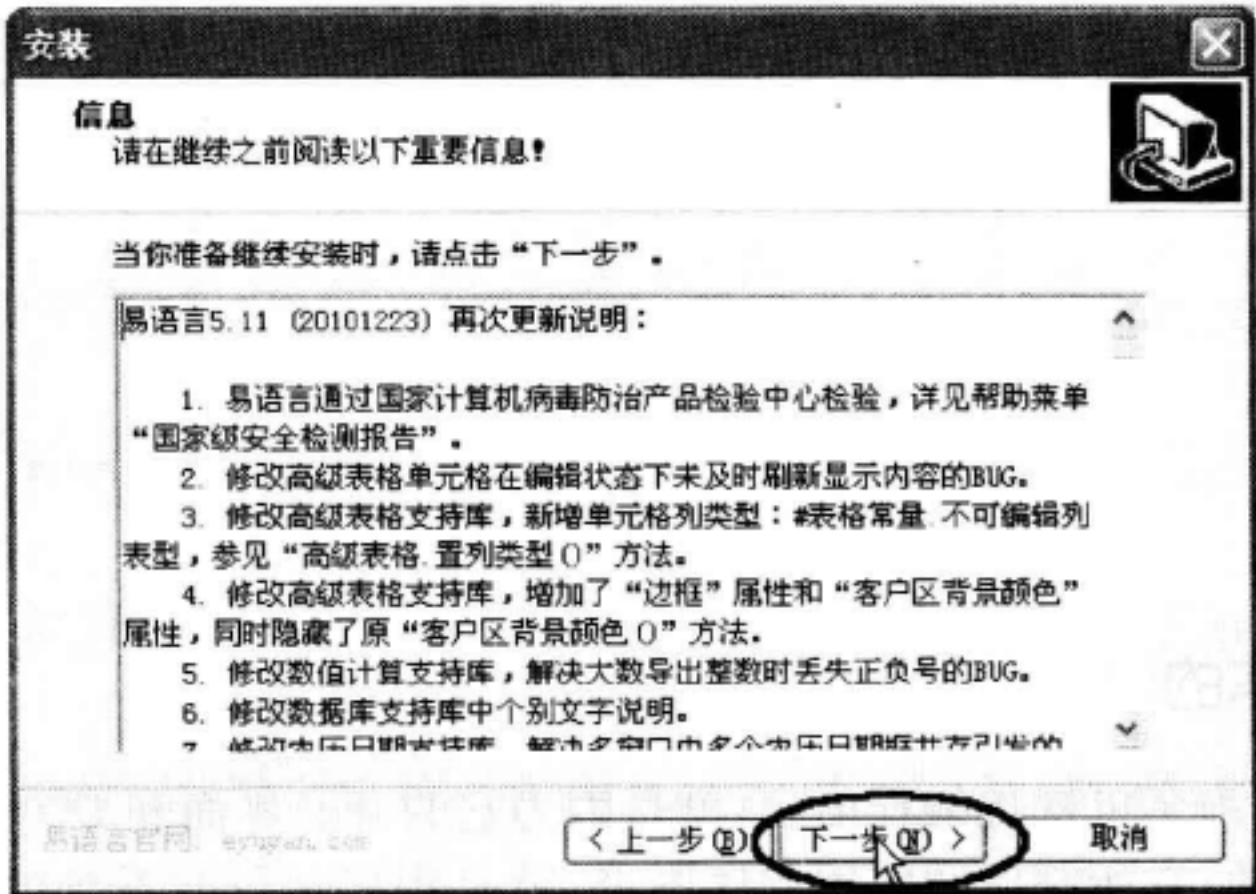


图 1-50 自述文件

步骤 8:弹出安装目录选择窗口,如图 1-51 所示。单击“浏览”选择安装的目标目录,完成后单击“下一步”。

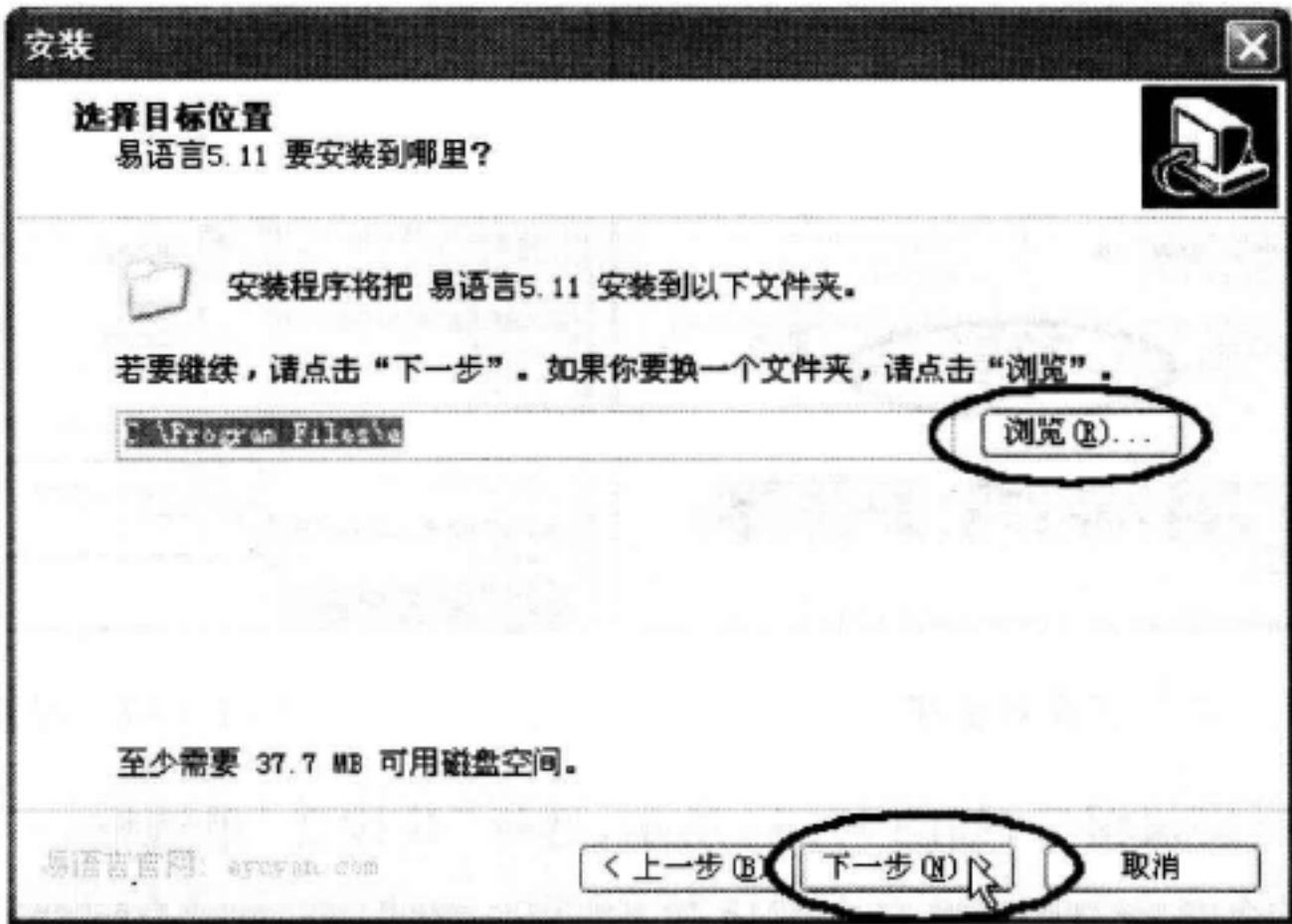


图 1-51 选择目标位置

步骤 9:安装到目标目录的所有文件如图 1-52 所示。其中 e.exe 是易语言主程序,ecom 文件夹用于存放易模块,help 目录用于存放帮助文件,lib 目录用于存放易语言支持库,samples 用于存放易语言的一些实例,tools 目录用于存放编程相关工具软件,readme.txt 用于记录新版本的信息。



图 1-52 安装成功

1.3 易语言基本界面操作

1.3.1 易语言的界面

易语言之所以称为全可视化编程语言,就是因为它的开发界面和 DOS 界面下的一般编程界面相比,是图形化的;它所制作的应用程序图形,就是程序运行时看到的程序界面。整个编程过程都是可视的,可以称之为“所见即所得”。

初次运行易语言后,首先会弹出对话框,询问创建何种类型的易程序,如图 1-53 所示。

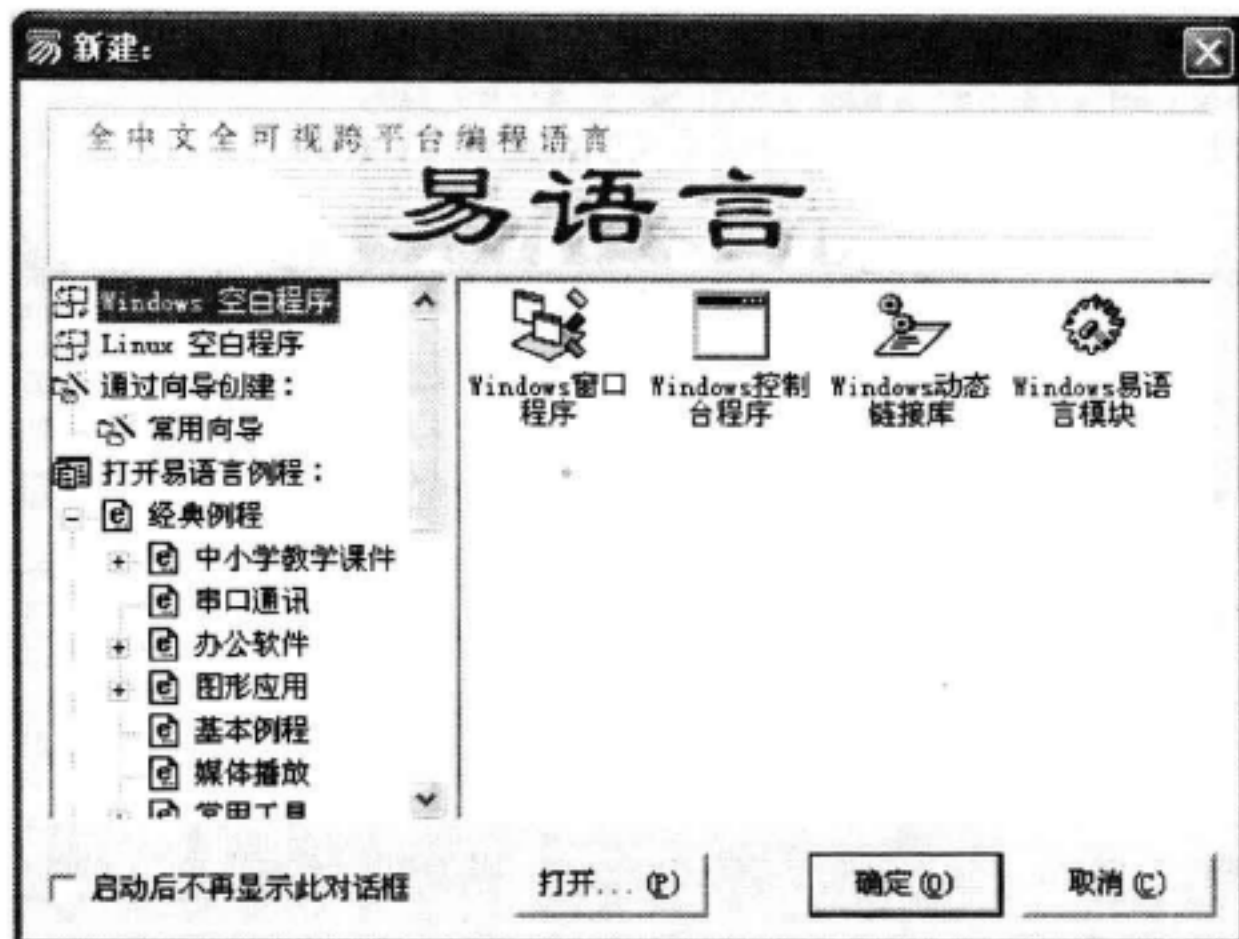


图 1-53 易语言启动对话框

若打开易语言界面后未新建程序,也可以通过菜单“程序”→“新建”来创建新的易程序。或点击窗口工具条中的新建按钮来新建易程序。

易语言可创建以下 6 种程序:

- Windows 窗口程序:是支持在 Windows 下弹出窗口及组件等标准 WIN32 位程序,也称易程序。
- Windows 控制台程序:是 WIN32 位无窗口界面的易程序。一般用于服务器等。
- Windows 动态链接库:可以生成 DLL 程序。
- Windows 易语言模块:简称易模块,是经过初步编译后的程序模块,供其他程序重复调用。
- Linux 控制台程序:是支持 Linux 操作系统的无窗口命令程序。
- Linux 易语言模块:是支持 Linux 操作系统且经过初步编译后的程序模块。

选择“Windows 窗口程序”,点击“确定”按钮,就会创建一个相应的标准的 Windows 窗口程序,并可以看到易语言的主界面,如图 1-54 所示。

界面看似有点复杂,但如果接触过 word 等办公软件或网页编辑工具,易语言的界面是不难理解的,因为里面的元素都是差不多的。程序窗口由上往下依次是标题栏、菜单栏和工具栏。这 3 个栏和所有的 Windows 应用程序都是一样的。

易语言主界面的最上方是标题栏,显示易语言版本及当前打开的程序名称,当前窗口名称,以及当前所支持的操作系统。标题栏下方是菜单栏,有易语言的常用菜单。菜单栏下方是快捷命令按钮工具条,一些常用的操作都可以通过点击这些工具条中的按钮实现。

主界面的左边是易语言的工作夹,其中有 3 个面板,分别是“支持库面板”、“程序面板”和“属性面板”。

“支持库面板”的作用是显示支持库列表,展开查看各支持库提供的命令、数据类型等信息。在程序编辑状态下,可以通过双击此面板中的某个命令,将其直接填充到光标处。若有窗口组件的方法也可以再找个列表中查看方法的用处。将光标移至某支持库根部,按下 F1 后可查看此支持库的介绍信息。

“程序面板”的作用相当于一个组织结构,可以添加窗口,或加载全局变量、常量、资源、

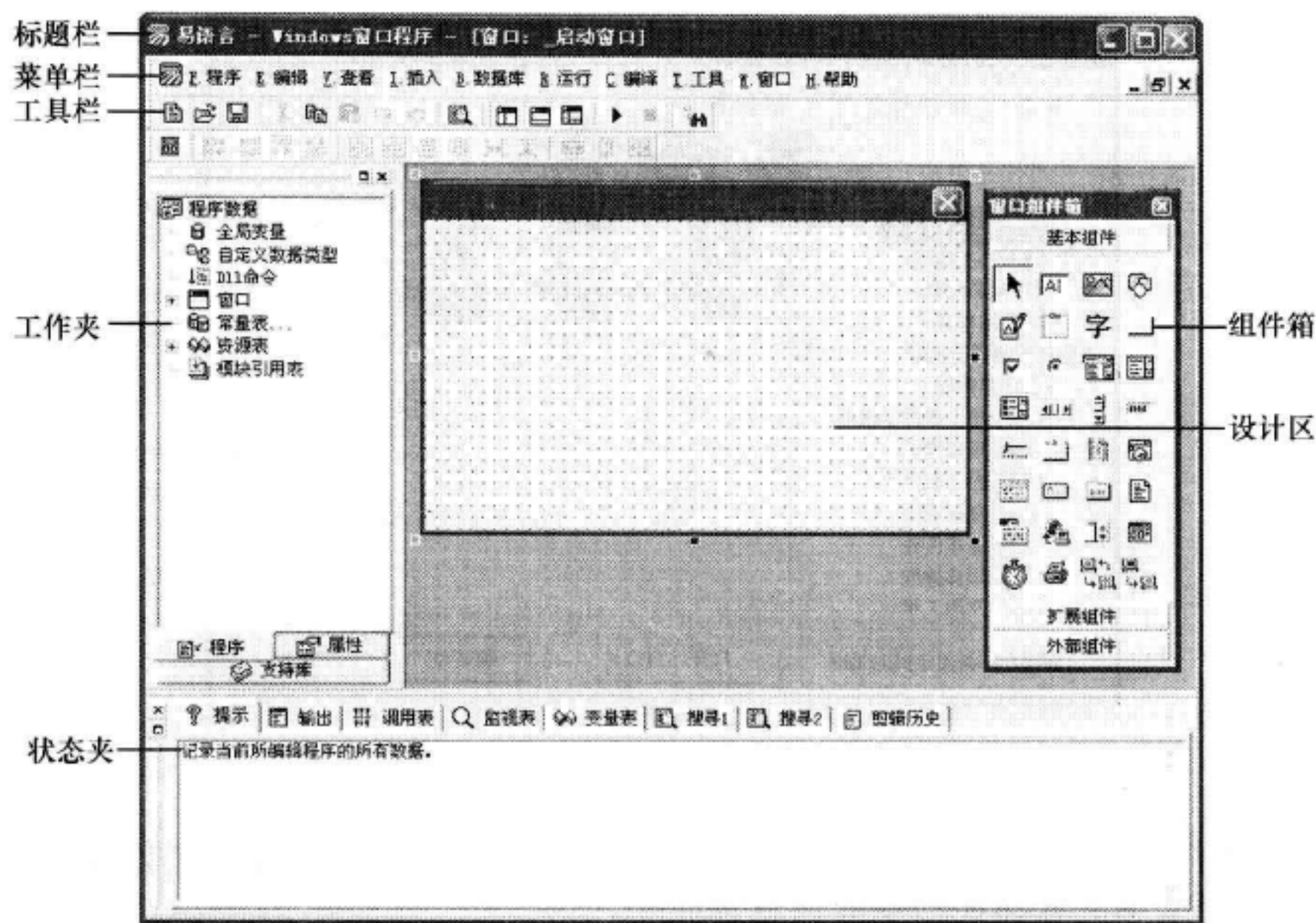


图 1-54 易语言主界面

DLL 命令申明、自定义数据类型等。也可用来在程序各操作界面间进行切换,例如可以直接找到某个创建的窗口中,或快速找到某个子程序。

“属性面板”的作用是用来查看和更改已添加组件的属性、组件列表列出所有组件并可快速选择所需组件。在属性面板的顶部有一个组合框,用作选择当前设计窗口中的某窗口单元。在属性面板的底部也有一个“事件组合框”,用作自动加入或跳转到当前被选择窗口单元的某事件处理子程序。

最右边是易语言的组件箱,又称为“窗口单元工具箱”、“控件工具箱”,这是一个非常重要的窗口,里面列出了易语言提供的所有组件。分为四栏,“基本组件”栏可显示易语言最基本常用的组件,即核心支持库内的组件;“扩展组件”包含扩展支持库内的组件;“外部组件”包含 COM 包装支持库所封装的 ActiveX 组件,此组件也称 OCX 组件;“外部事件组件”包含 COM 包装支持库所封装的 COM 事件组件。

主界面中间的窗口是设计区,称为“界面窗口”,也称之为“设计窗口”,在窗口设计时可自由向窗口中添加组件,进行程序界面设计;在程序代码编辑状态下可录入、修改程序代码。切换这两个工作状态可通过“窗口”菜单或“程序面板”等实现。

最下方是易语言的状态夹,可以查看帮助信息,查看调试文本等。在状态夹中包括 3 个子夹,分别是“提示”、“输出”和“调用表”。

“提示夹”中始终具有针对于当前用户操作的提示,比如在“属性夹”里点击“高度”属性,在“提示夹”里回出现提示:“本属性记录当前窗口高度,单位为像素点”。

“输出夹”中提供当前系统输出给用户的信息。比如编译易程序的全过程。

“调用表夹”中提供调试时当前被中断运行的易程序的子程序调用过程记录。

新建一个程序之后,首先进入的画面为“界面编辑窗口”,若想切换到“代码编辑窗口”,选中并双击中间的窗口即可切换到“代码编辑窗口”,如图 1-55 所示。

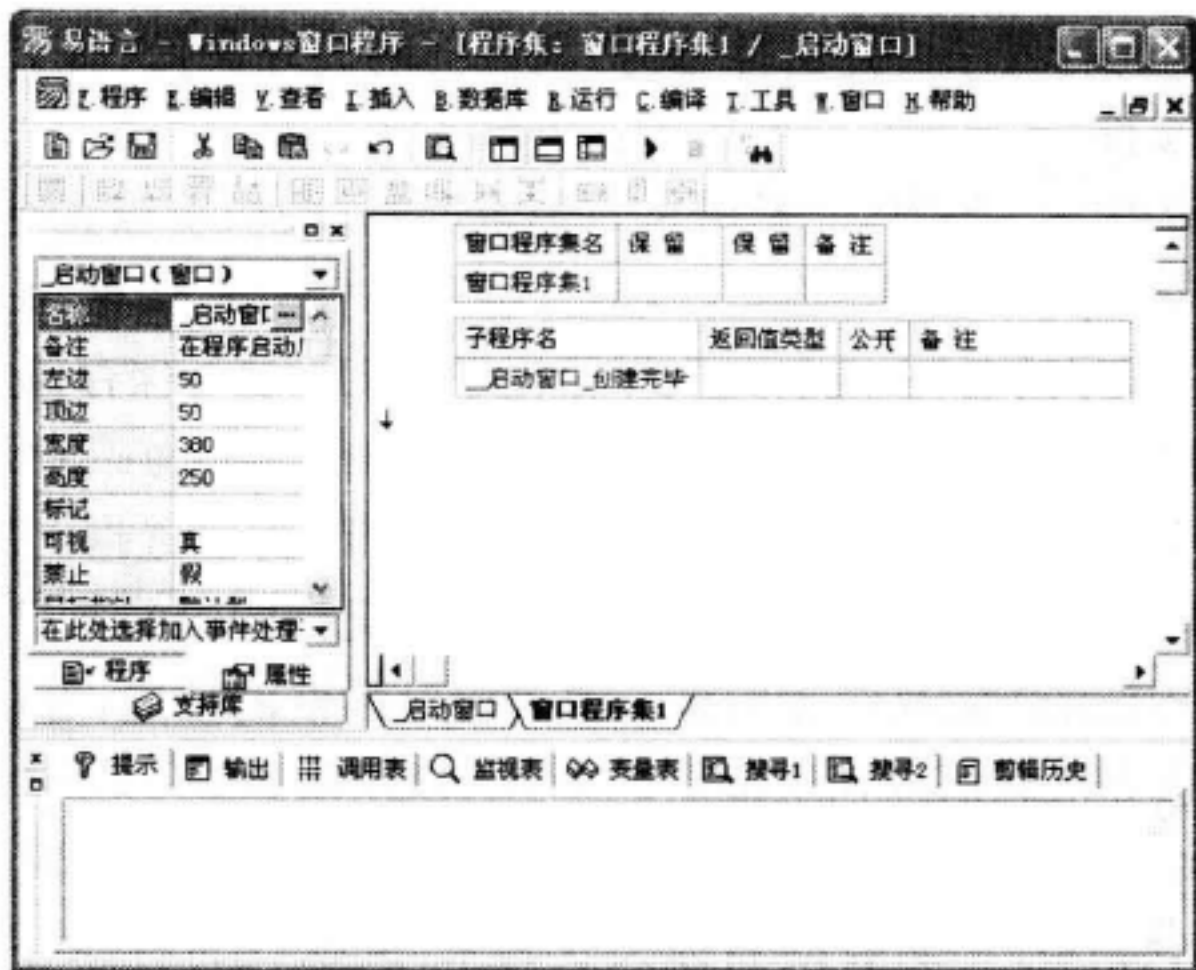


图 1-55 代码编辑窗口

1.3.2 易语言的菜单栏

易语言共有 10 个菜单项,如图 1-56 所示。



图 1-56 菜单栏

1. “程序”菜单

相当于其他应用程序的“文件”菜单。这个菜单主要负责文件操作,是程序操作相关的功能集合,例如“新建”、“打开”、“保存”、“另存为”等,还包括“程序配置”、“易模块管理”等,如图 1-57 所示。

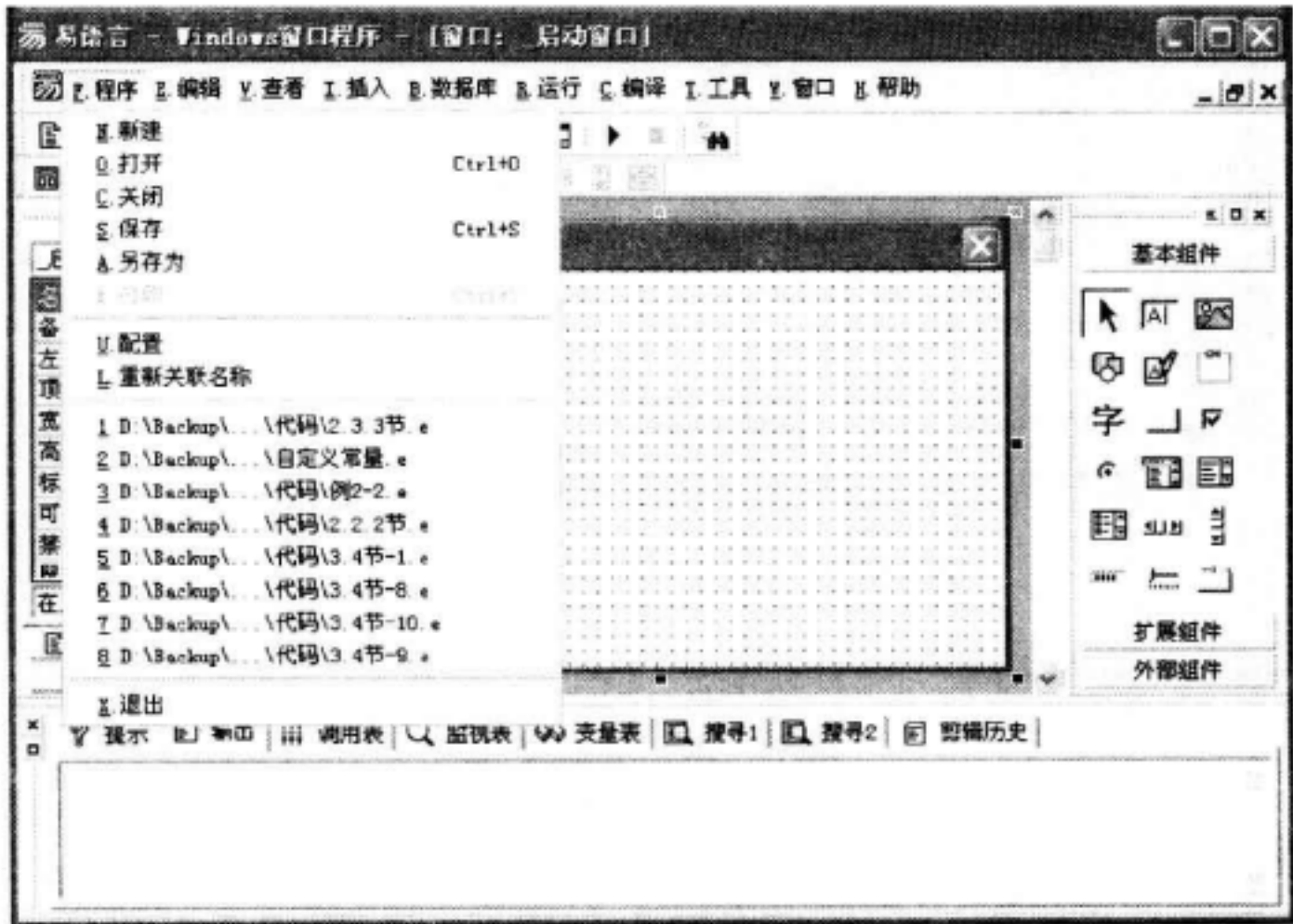


图 1-57 程序菜单

程序菜单中各个选项的说明如下:

- 新建:建立新程序。弹出标题为“新建:”的对话框,可以选择创建不同类型的程序。

- 打开:打开一个现有程序。弹出标题为“请选择易程序文件:”的打开文件对话框,选择后缀为“*.e”的程序文件。
- 关闭:关闭当前程序。关闭后“易语言”的程序设计窗口将被置空。
- 保存:保存当前程序。如果新建程序没有保存过,将弹出“保存为:”对话框,提示编辑者选择程序的保存位置和程序的名称,此后再次保存程序将默认这个保存位置,不会再弹出提示。
- 另存为:将当前程序以一个新文件名保存。将弹出“另存为:”对话框,提示编辑者选择程序的新的保存位置或输入程序的新的名称进行保存,同时将位置指向新保存的程序。
- 打印:打印当前编辑窗口中的源程序。使用打印机打印当前窗口中的源代码。
- 配置:配置本程序的环境及作者信息。通过该对话框的位置,可以将程序名称、程序描述、程序备注、作者信息等信息保存在生成后的 EXE、以模块、DLL 文件中,当查看此文件的属性时,这些信息会显示处理。同时可以在“设置程序图标”中为程序设置图标。
- 重新关联名称:将程序中所有的名称关联起来,关联后的名称可以被系统内置名称管理器自动跟踪处理。
- 退出:退出系统,提示保存文档。被更改过或未被保存过的程序,将弹出信息框提示编辑者保存程序,然后退出系统。在“重新关联名称”和“退出”之间是最近打开 iade 程序,可用鼠标左键单击打开被选择程序,同时原有程序被关闭。

2. “编辑”菜单

和其他应用程序的编辑菜单一样,这个菜单集成了所有的文本编辑命令,例如“剪切”、“复制”、“粘贴”、“查找”等,还包括“菜单编辑器”,用来编制程序的菜单。如图 1-58 所示。“编辑”菜单主要是针对代码编辑窗口而言的。

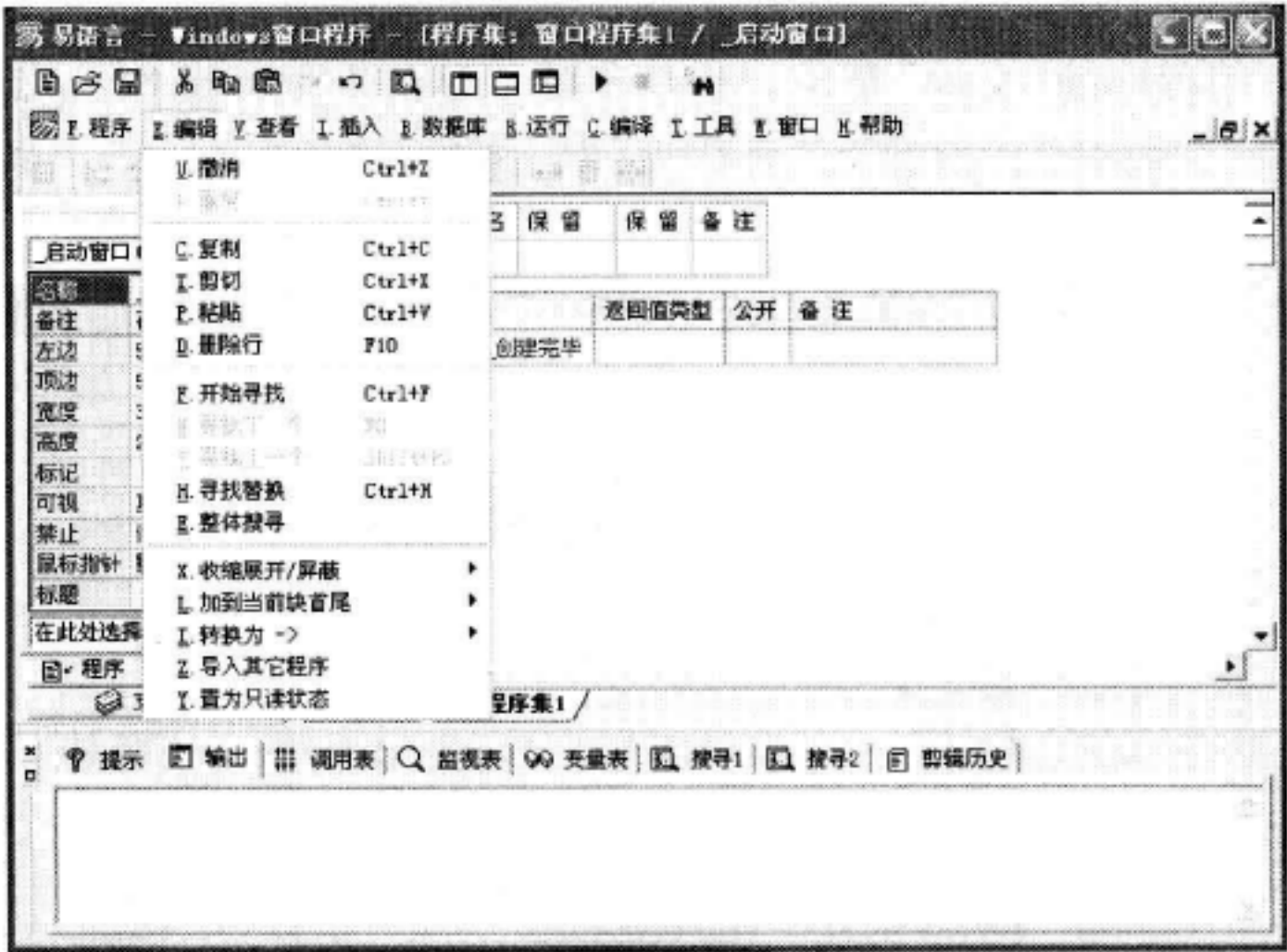


图 1-58 编辑菜单

编辑菜单中各选项的说明如下:

- 撤销:撤销最后一步操作。一步步撤销自创建或打开程序后对程序的每一次修改。该功能的快捷键为:Ctrl + Z。
- 重复:重新执行还原先前已撤销的操作。一步步还原自程序被创建或打开后撤销的操作。该功能的快捷键为:Ctrl + Y。

- 复制:复制被选内容并将其置于系统剪辑板中。复制被选中代码或窗体、窗体组件到系统剪辑板,其原有的内容不会改变。该功能的快捷键为:Ctrl + C。
- 剪切:剪切被选内容并将其置于系统剪辑板中。相当于将被选中代码或窗体、窗体组件剪切到系统剪辑板中,其原有内容被删除。该功能的快捷键为:Ctrl + X。
- 粘贴:将系统剪辑板内容插入到当前位置。将系统剪辑板的内容插入到程序中。如果剪辑板内容是程序代码,需要在代码设计区中进行插入;如图内容是窗体组件,需要选中欲作为容器的窗口或窗口组件才能插入;如果是窗体,只需要激活易语言系统,就可以将窗体插入到程序中。该功能的快捷键为:Ctrl + V。
- 删除行:删除当前所选择的区域或光标当前所在的行。该功能的快捷键为:F10。
- 开始寻找:开始在程序中寻找指定文本。弹出“寻找对话框”,请求输入被寻找的文本。其寻找范围为当前程序集。该功能的快捷键为:Ctrl + F。
- 寻找下一个:在程序中寻找下一个指定文本。以光标或已寻找到的文本为界,向代码下方寻找。其寻找范围为当前程序集。该功能的快捷键为:F3。
- 寻找上一个:在程序中寻找上一个指定文本。以光标或已寻找到的文本为界,向代码上方寻找。其寻找范围为当前程序集。该功能的快捷键为:Shift + F3。
- 寻找替换:在程序中寻找并替换指定的文本。弹出“寻找替换对话框”,提示输入被替换和替换成的文本。以光标或已寻找到的文本为界,向下寻找或替换文本,也可以将当前程序集中所有找到的指定文本进行替换。该功能的快捷键为:Ctrl + H。
- 整体搜寻:在程序中寻找指定文本并列出所有找到的项目。在全局中寻找指定文本,包括常量数据表、数据类型表等所有在代码设计区中以文本形式存在的指定项目。
- 收缩展开/屏蔽:将当前子程序或块内的所有语句收缩以“*** 缩略程序块 ***”显示,如图 1-59 所示。

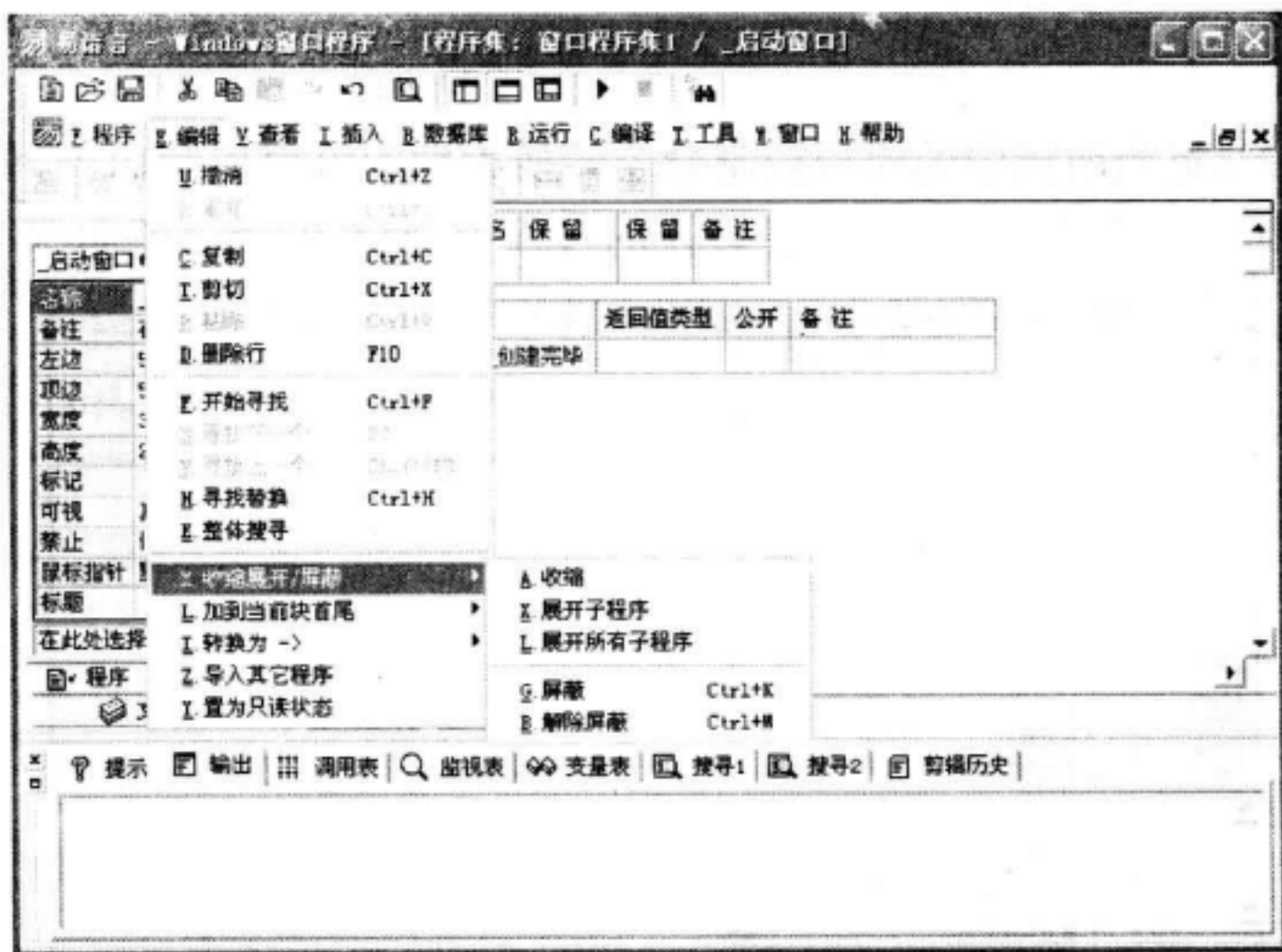


图 1-59 收缩展开/屏蔽子菜单

- 收缩:将当前选中的子程序或块内的所有语句收缩显示,如果需要打开可以直接单击缩略子程序或语句左边的加号图标。
- 展开子程序:将当前选中块内的所有被收缩子程序展开显示。

- 展开所有子程序:将当前程序中所有被收缩子程序展开显示。
- 屏蔽:屏蔽当前所选中的代码块。把所选代码行或代码段设置为注释,使其在调试和运行程序时不被执行。该功能的快捷键为:Ctrl + K。
- 解除屏蔽:解除屏蔽当前所选中的代码块。把注释行或被屏蔽的代码设置为可执行代码。该功能的快捷键为:Ctrl + M。
- 加到当前块首尾:将流程控制命令加入到当前所定义程序块的首部和尾部,如图 1-60 所示。添加一个新的流程控制命令,并将被选择代码块放到此命令中。要使得菜单功能有效,选择代码块的方法为:选择两行或两行以上的单行代码;选择一个或多个分支流程控制命令,必须把流程线外的一行选中。

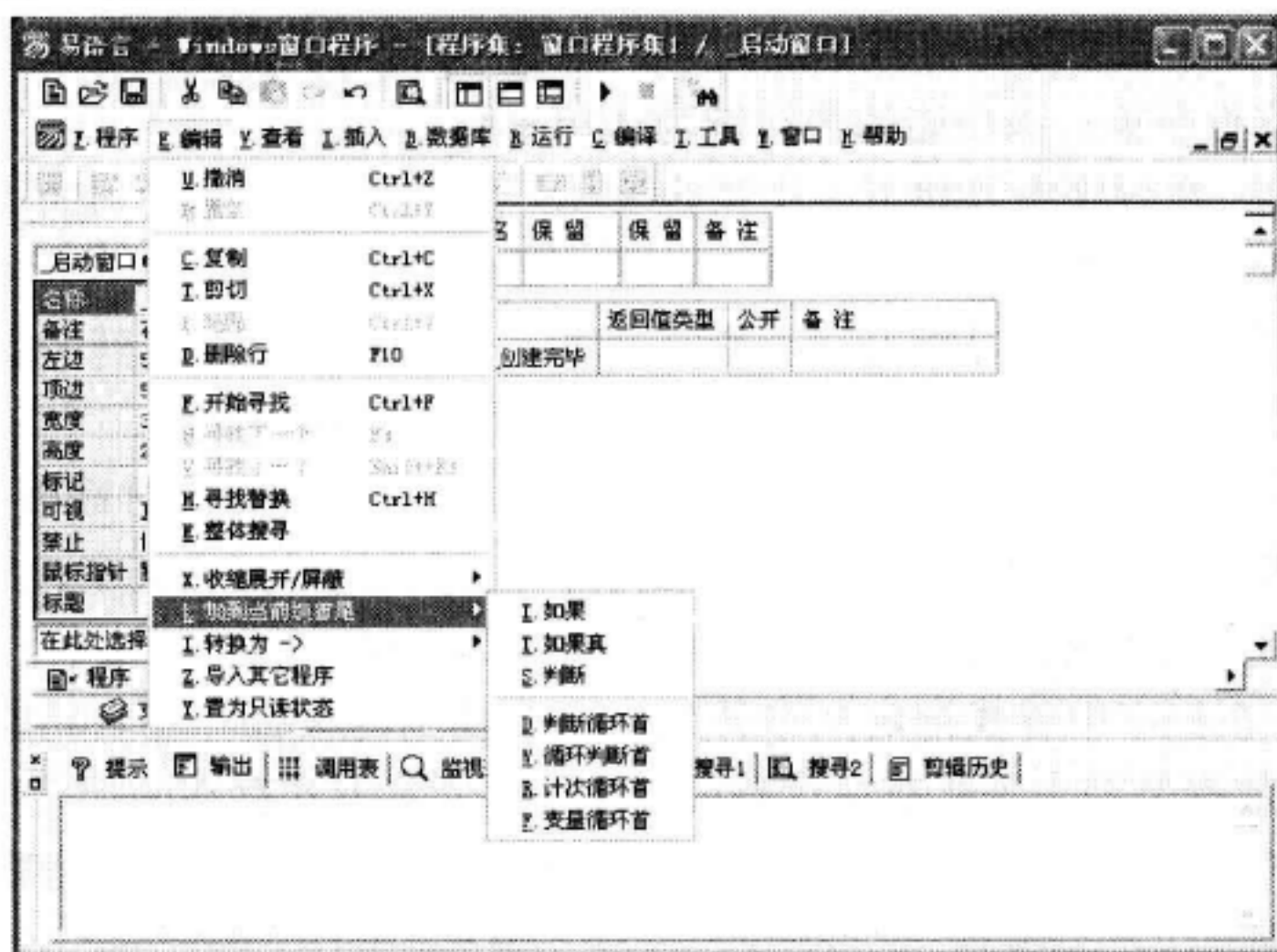


图 1-60 加到当前块首尾子菜单

- 转换为 ->:将当前选中的流程控制命令转换为别的流程控制命令,如图 1-61 所示。它们之间可以互相转换,但要注意的是,在转换的过程中,原来的程序流向有可能发生改变。

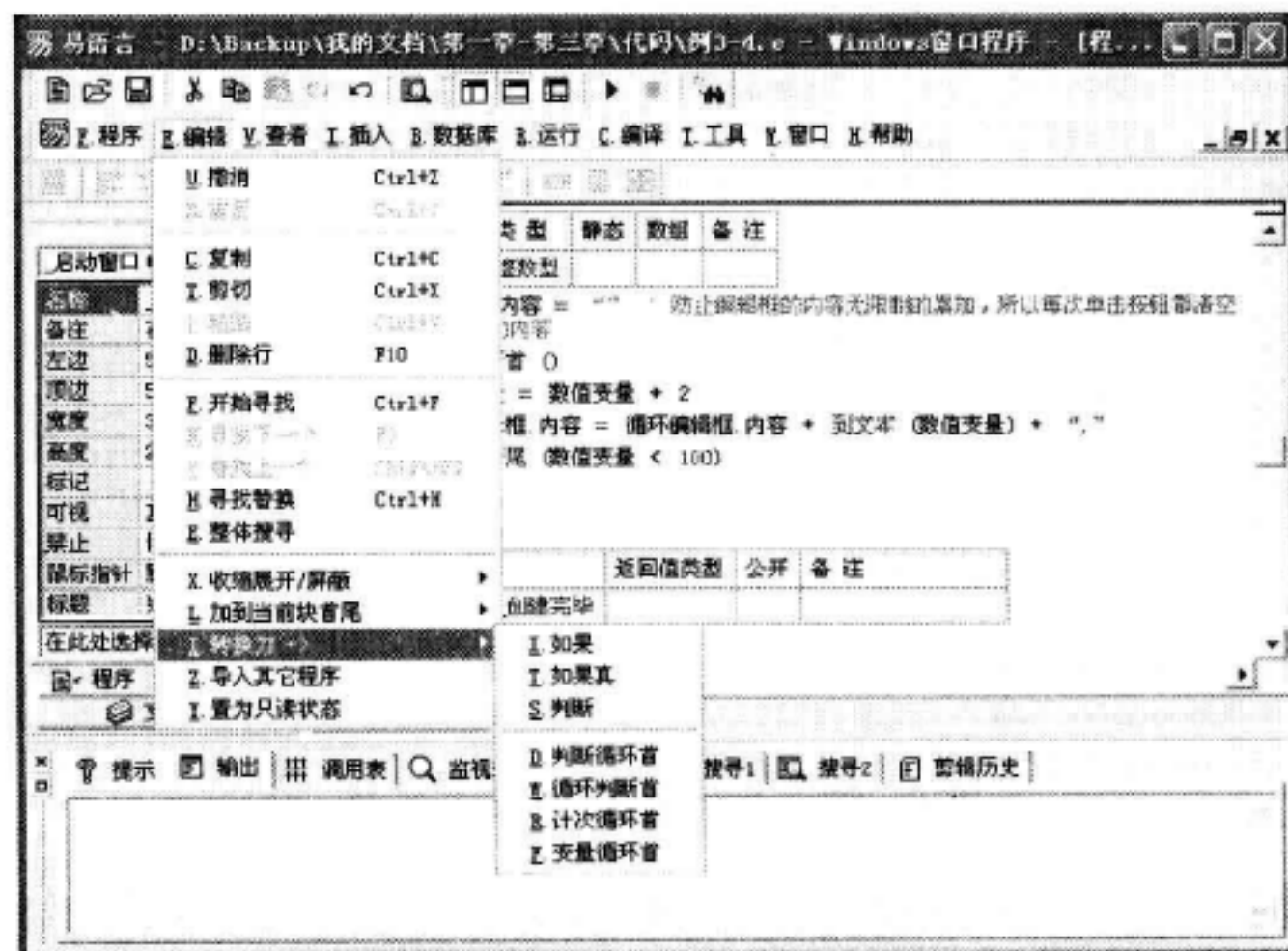


图 1-61 转换为子菜单

- 导入其他程序:将其他易程序中的内容全部导入到本程序中。打开标题为“请选择易程序文件:”对话框,选择程序文件插入到当前程序中。被导入程序的“_启动窗口”以及其他与当前程序重复的程序集名称、窗口名称后面将按顺序被加入数字加以区别。
- 置为只读状态:设置为只读状态后将不允许所有修改操作的发生。

3. “查看”菜单

“查看”菜单包含以下几类功能:控制工具条的显示;跳转到程序内各种资源的定义处;进行代码快速定位;快速查看某窗口运行时的显示效果,如图 1-62 所示。主要用于对程序的表格型代码部分进行访问,如“自定义数据类型表”、“全局变量表”、“资源表”等。



图 1-62 查看菜单

查看菜单各选项说明如下:

- 工具条:在易语言界面上显示和隐藏各功能工具,如图 1-63 所示。

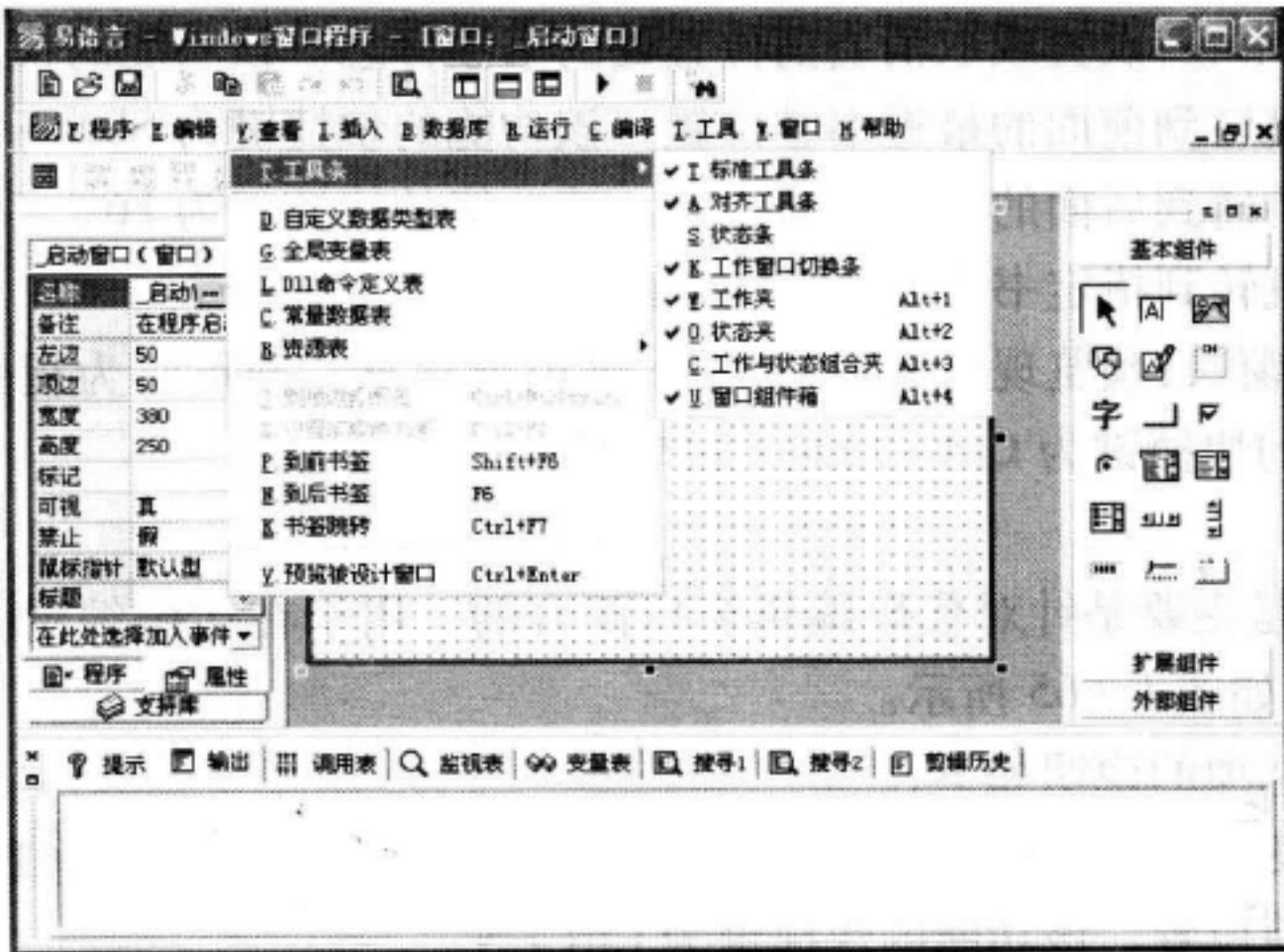


图 1-63 查看菜单

- 自定义数据类型表:跳转到自定义数据类型定义表处。
- 全局变量表:跳转到全局变量定义表处。

- Dll 命令定义表:跳转到 Dll 命令定义表处。
- 常量数据表:跳转到常量数据定义表处。
- 资源表:跳转到资源表处,与工作夹中程序面板中对应项目功能相同,如图 1-64 所示。



图 1-64 资源表子菜单

- 到最近修改处:跳转到最近修改的位置。该功能的快捷键为 Ctrl + BackSpace (退格键)。
- 设置或取消书签:设置或取消当前位置处的书签标志。该功能的快捷键为 Ctrl + F6。
- 到前书签:跳转到前面的最近书签位置。该功能的快捷键为 Shift + F6。
- 到后书签:跳转到后面的最近书签位置。该功能的快捷键为 F6。
- 书签跳转:跳转到指定书签处。该功能的快捷键为 Ctrl + F7。
- 预览被设计窗口:预览现行窗口,按 Esc 键退出预览。预览时为程序编写的代码将不会被执行。该功能的快捷键为 Ctrl + Enter (回车键)。

4. “插入”菜单

“插入”菜单也是主要是针对代码编辑窗口而言的。用于在代码编辑窗口插入各种类型的程序成分和代码,如图 1-65 所示。

插入菜单中各选项的说明如下:

- 现行单元:根据现行编辑窗口的性质插入一个新子程序/数据类型/全局变量/Dll 命令/常量/资源到当前位置。该功能的快捷键为 Ctrl + N。
- 类模块:插入一个新类模块。
- 程序集:插入一个新程序集。
- 子程序:插入一个新的子程序到当前位置的后面。
- 窗口:插入一个新窗口。

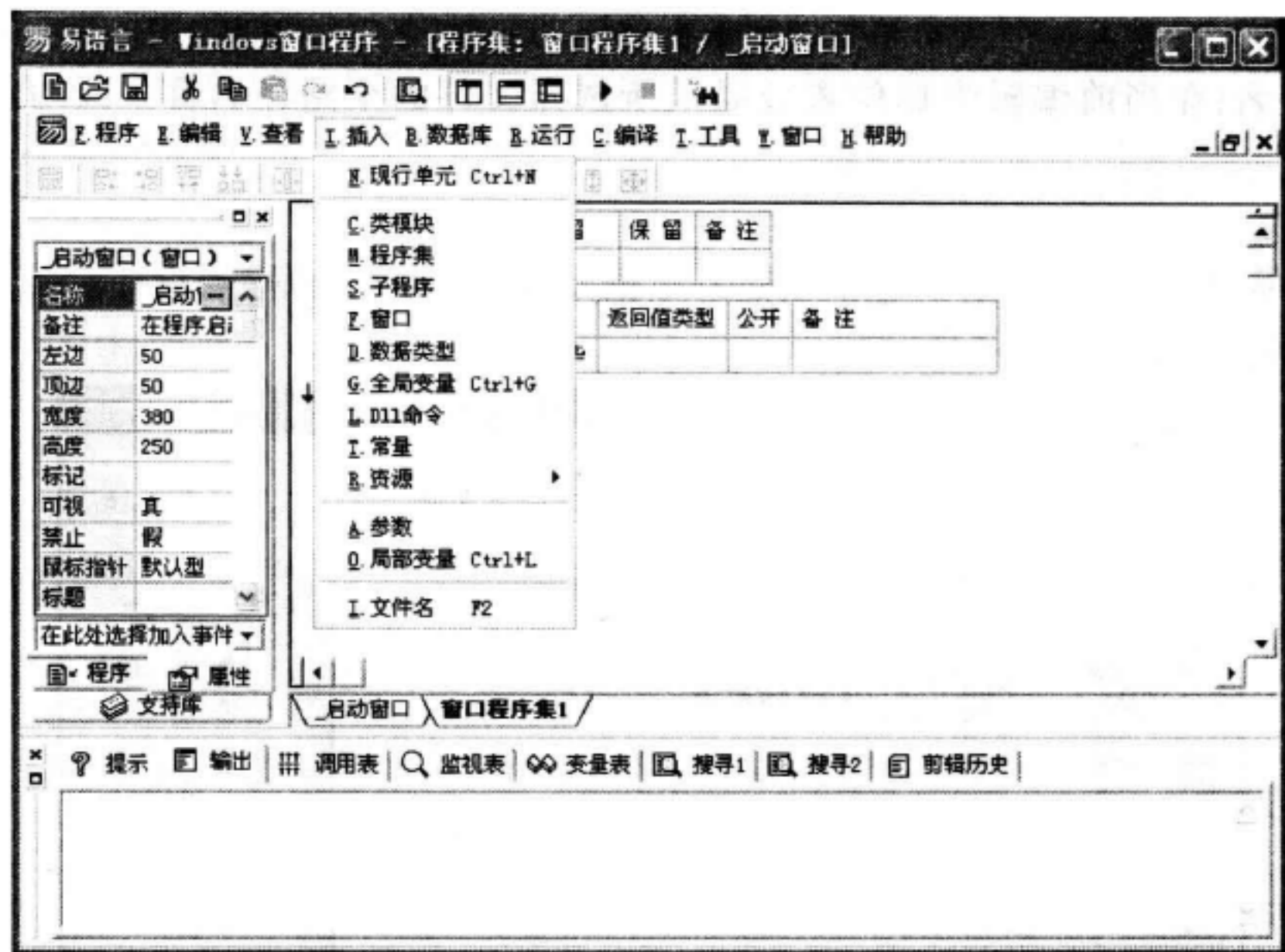


图 1-65 插入菜单

- 数据类型:插入一个新的自定义数据类型到数据类型表。
- 全局变量:插入一个新的全局变量到全局变量表。该功能的快捷键为 Ctrl + G。
- DLL 命令:插入一个新的 DLL 命令到 DLL 命令表。
- 常量:插入一个新的常量到常量数据表。
- 资源:向资源表中添加数据资源。如文本文件、声音图片文件以及其他类型文件,如图 1-66 所示。

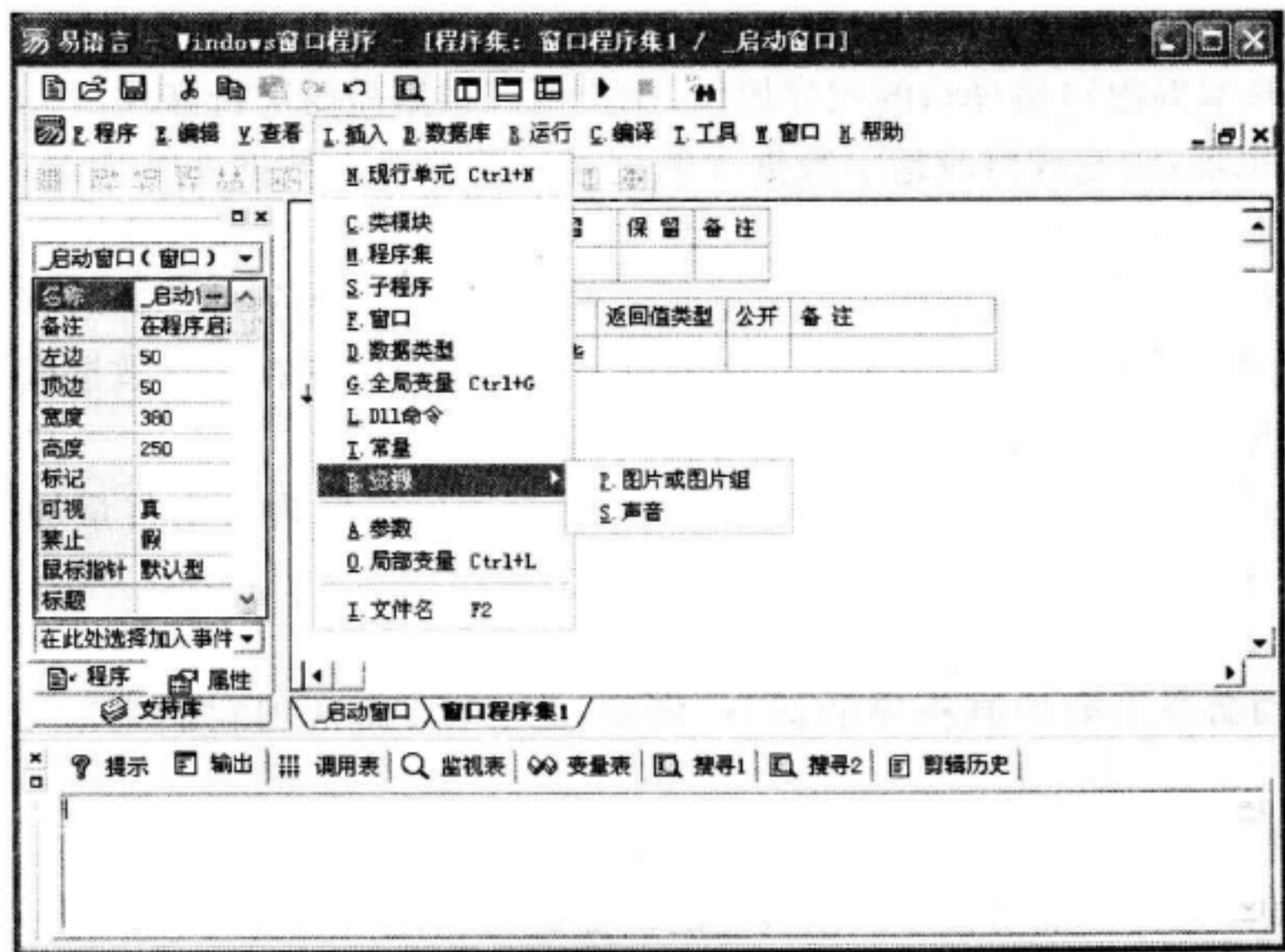


图 1-66 资源子菜单

- 参数:插入一个新参数到程序中当前命令或子程序调用的参数表。

- 局部变量:插入一个新局部变量到子程序局部变量表。该功能的快捷键为 Ctrl + L。
- 文件名:在当前编辑光标位置处插入所选择文件的全路径名称。该功能的快捷键为 F2。

5. “数据库”菜单

用于对易数据库进行操作。包括创建易数据库的“结构编辑器”和直接编辑易数据库的“记录编辑器”等,如图 1-67 所示。

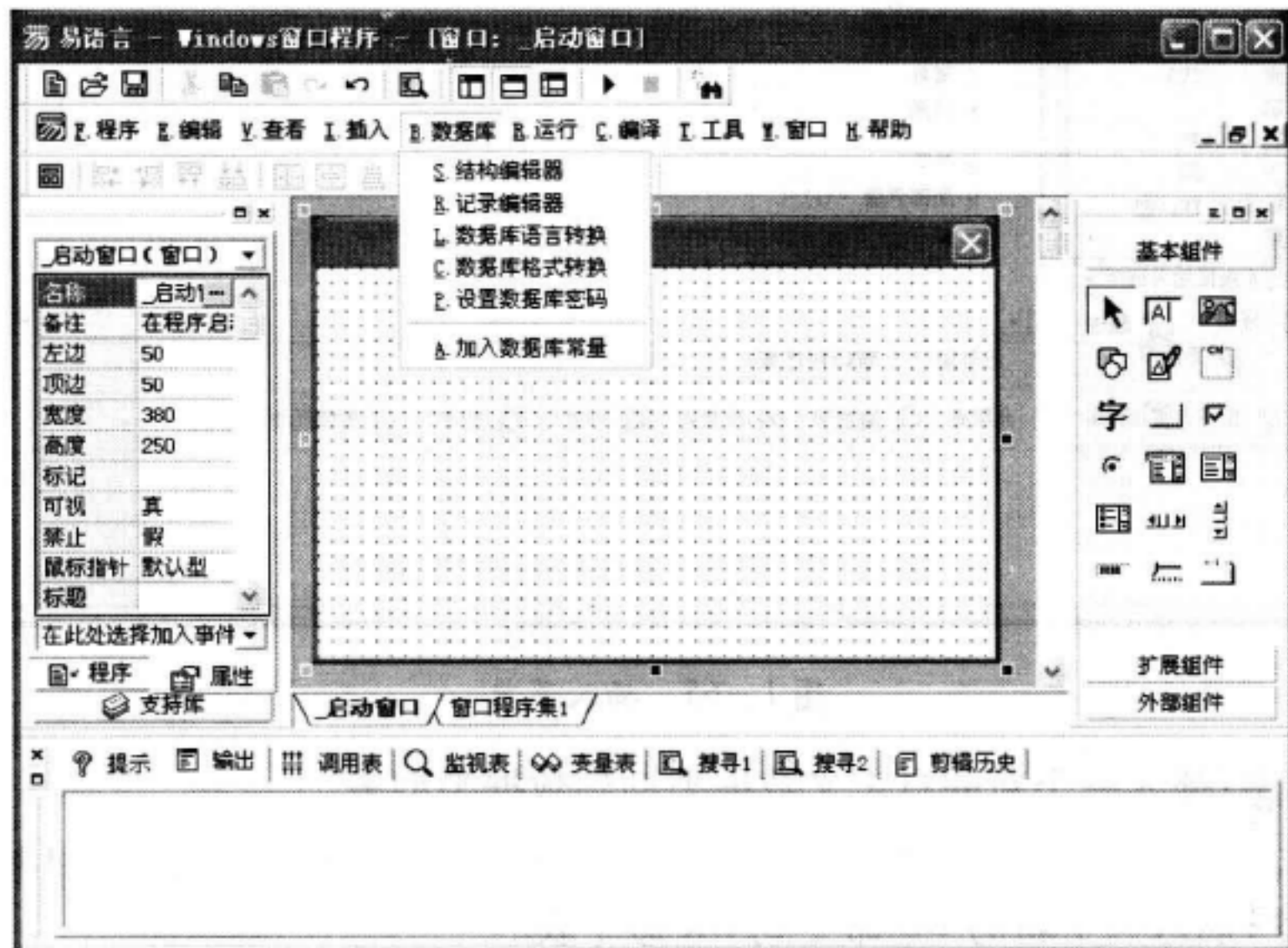


图 1-67 数据库菜单

数据库菜单中各选项的说明如下:

- 结构编辑器:浏览或修改设计数据库的结构。该功能由数据库管理器 .e (易语言安装目录\samples\基本例程\) 编译后的程序提供,用户可以根据需要进行修改。
- 记录编辑器:浏览或修改指定数据库的记录。该功能由数据库管理器 .e 编译后的程序提供,用户可以根据需要进行修改。
- 数据库语言转换:将数据库中的数据所使用的语言转换为另外一种。
- 数据库格式转换:可以将其他类型的数据库通过 ODBC 转换为易数据库。
- 设置数据库密码:设置指定数据库的访问密码。
- 加入数据库常量:将指定数据库的名称及所有字段名作为文本常量加入到系统常量表,以便在程序中使用。

6. “运行”菜单

这个菜单的命令主要控制程序的运行、暂停、终止等,通常用它们来演示程序效果,如图 1-68 所示。

运行菜单中各选项的说明如下:

- 运行:调试运行当前易程序。该功能的快捷键为 F5。
- 终止:终止现行调试运行的易程序。该功能的快捷键为 Ctrl + F5。
- 查看表达式/变量:查看/修改指定表达式或变量的内容。该功能的快捷键为 Shift + F9。

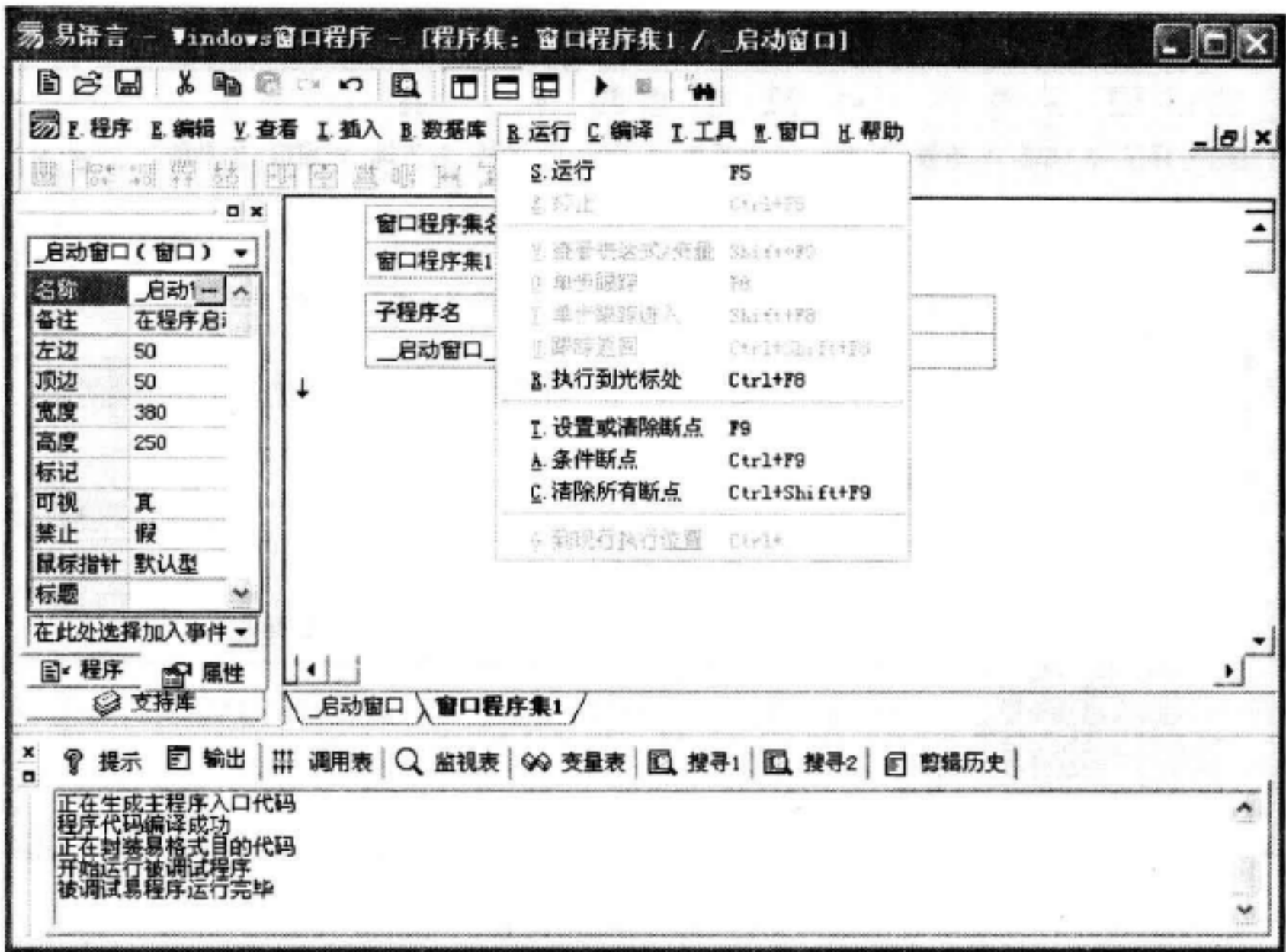


图 1-68 运行菜单

- 单步跟踪:在程序现行运行位置单步执行一程序,如果此程序调用了子程序,系统不会跟踪到该子程序中去。该功能的快捷键为 F8。
- 单步跟踪进入:在程序现行运行位置单步执行一程序,如果此程序行调用了子程序,则跟踪进入子程序。该功能的快捷键为 Shift + F8。
- 跟踪返回:跳出现行子程序(后面的代码会被执行),转到上级调用此现行子程序的语句后中断。该功能的快捷键为 Ctrl + Shift + F8。
- 执行到光标处:运行易程序,在当前光标所在程序行处中断。该功能的快捷键为 Ctrl + F8。
- 设置或清除断点:设置或清除当前程序行处的断点。该功能的快捷键为 F9。
- 条件断点:设置或修改当前程序行处的条件断点。该功能的快捷键为 Ctrl + F9。
- 清除所有断点:清除掉程序中的所有断点。该功能的快捷键为 Ctrl + Shift + F9。
- 到现行执行位置:跳到现行即将被执行语句的位置。该功能的快捷键为 Ctrl + .。

7. “编译”菜单

“编译”菜单是利用编译程序将易语言编写的源程序生成目标程序。包括“编译”、“静态编译”和“编译生成安装软件”,如图 1-69 所示。

其中编译出的 EXE 小巧,但是如果需要在没有易语言环境的计算机上使用,必须附带支持库。

而静态编译是编译器在编译可执行文件的时候,将可执行文件需要调用的对应动态链接库(.so)中的部分提取出来,链接到可执行文件中去,使可执行文件在运行的时候不依赖于动态链接库。简单地说静态编译虽然文件较大,但可以不依赖易语言系统文件,因此在没有易语言环境的计算机上仍然能使用。

编译生成安装软件是制作当前易语言程序的安装软件,该软件不依赖任何易语言系统文件,可以在未安装易语言系统的电脑上运行并安装指定易语言程序。

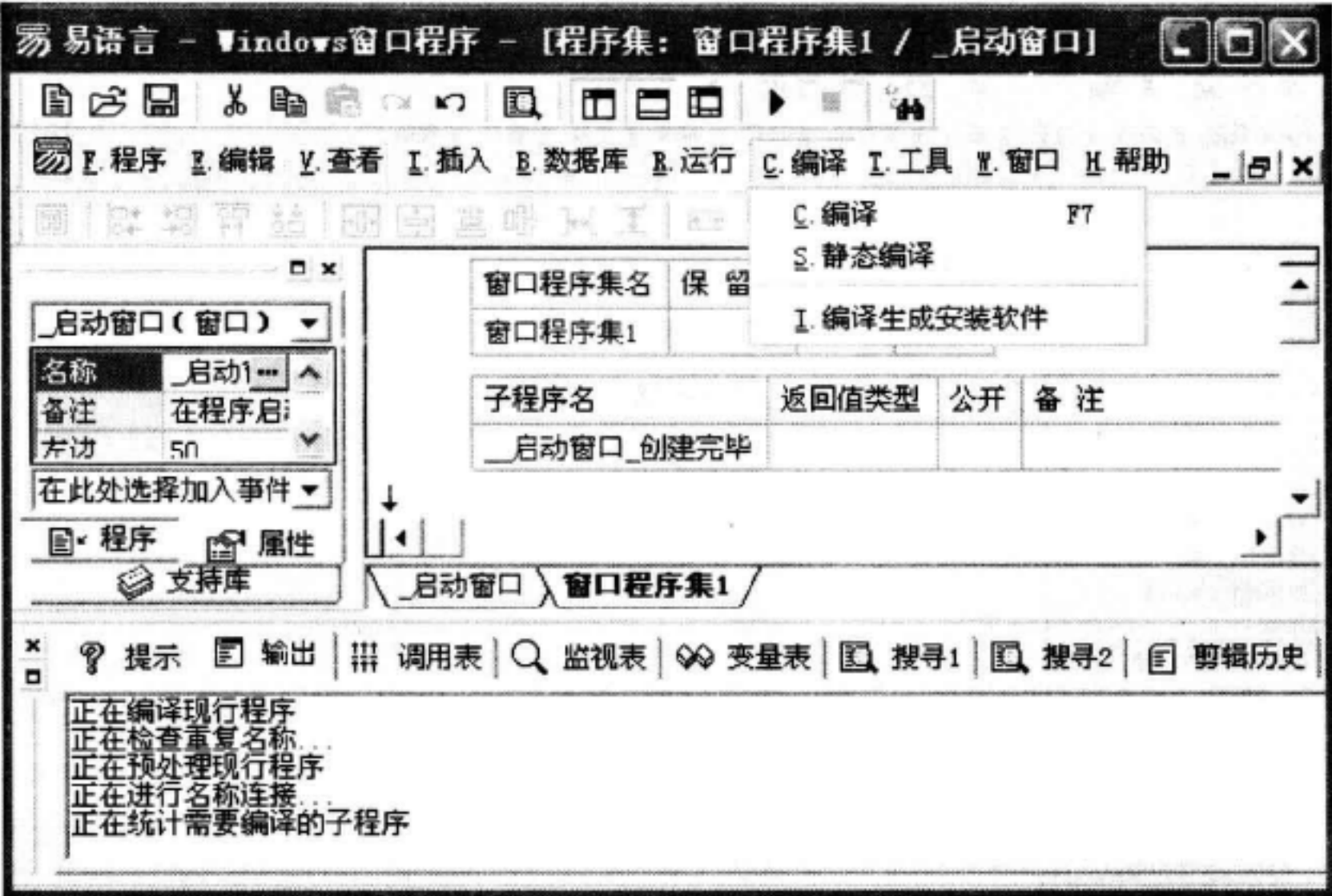


图 1-69 编译菜单

8. “工具”菜单

“工具”菜单包括易语言提供的多种附加工具,用来管理和配置易语言的扩展功能如“菜单编辑器”、“报表编辑器”、“支持库配置”、“系统配置”等工具选项,如图 1-70 所示。

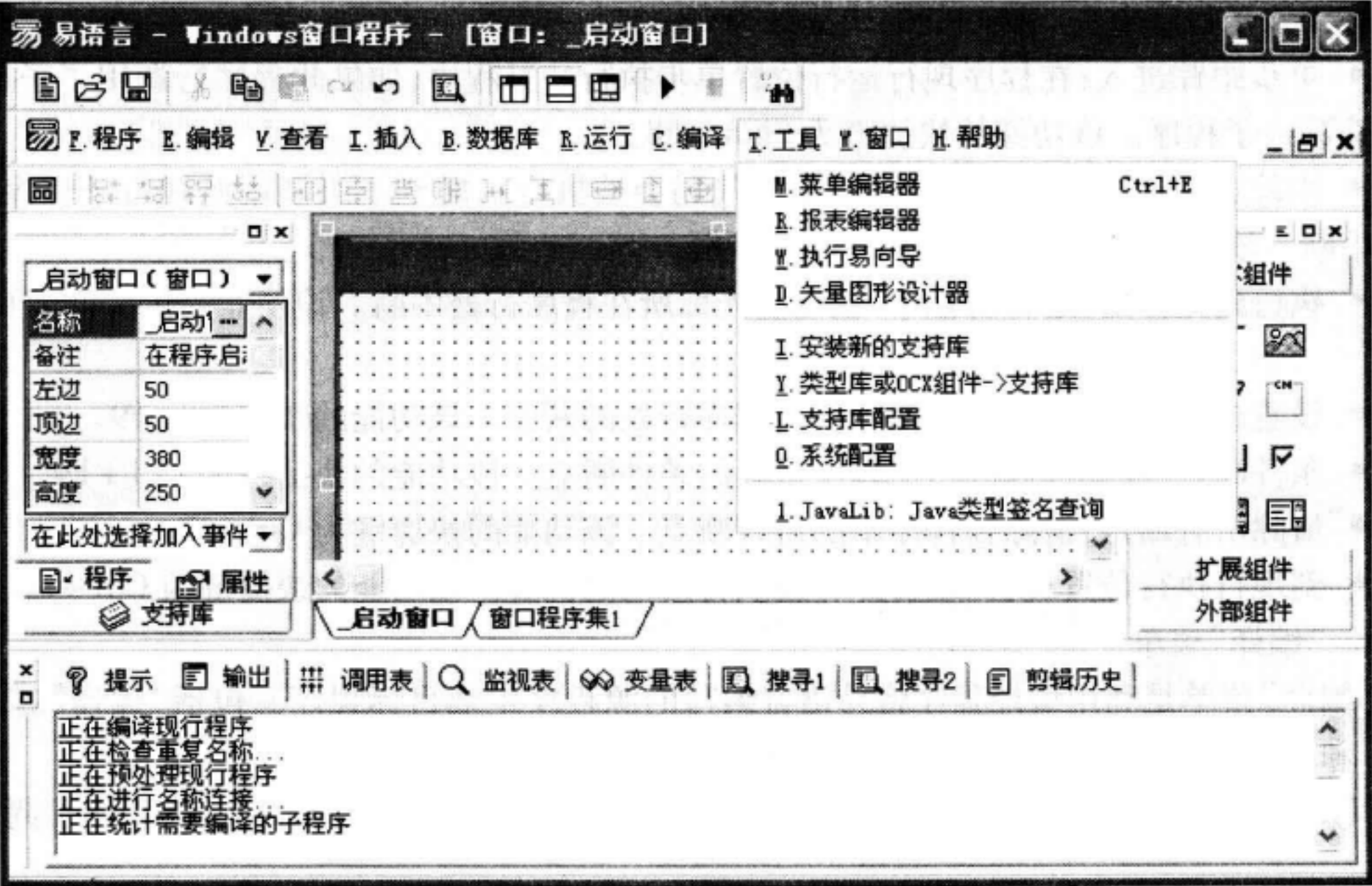


图 1-70 工具菜单

工具菜单中各选项的说明如下：

- 菜单编辑器:调用菜单编辑器编辑修改当前窗口的菜单。该功能的快捷键为 Ctrl + E。
- 报表编辑器:编辑报表模板文件。
- 执行易向导:执行指定的易向导文件。

- 矢量图形设计器:设计矢量图形供矢量图形设计器使用。
- 安装新的支持库:安装新的支持库或制作支持库安装包。
- 类型库或 OCX 组件→支持库:可以封装指定的 COM(组件对象模型)库或 OCX(对象链接和嵌入用户控件),使其能够在易语言中被使用。
- 支持库配置:易语言 3.8 以上版本初始安装后“支持库面板”默认只显示系统核心支持库,如果需要使用更多的支持库,需要在这里配置。
- 系统配置:根据个人习惯进行相关配置,通过调整该对话框中各选项属性的参数,可以自定义界面各部位颜色,也可以选择各种配色方案,还可以更改代码字体和对内置输入习惯等很多方面进行配置。

9. “窗口”菜单

和其他应用程序的窗口菜单一样,用于含有多个子窗口之间的互相切换,也可以选择子窗口的分布方式,水平/垂直平铺、层叠、排列图标等,如图 1-71 所示。

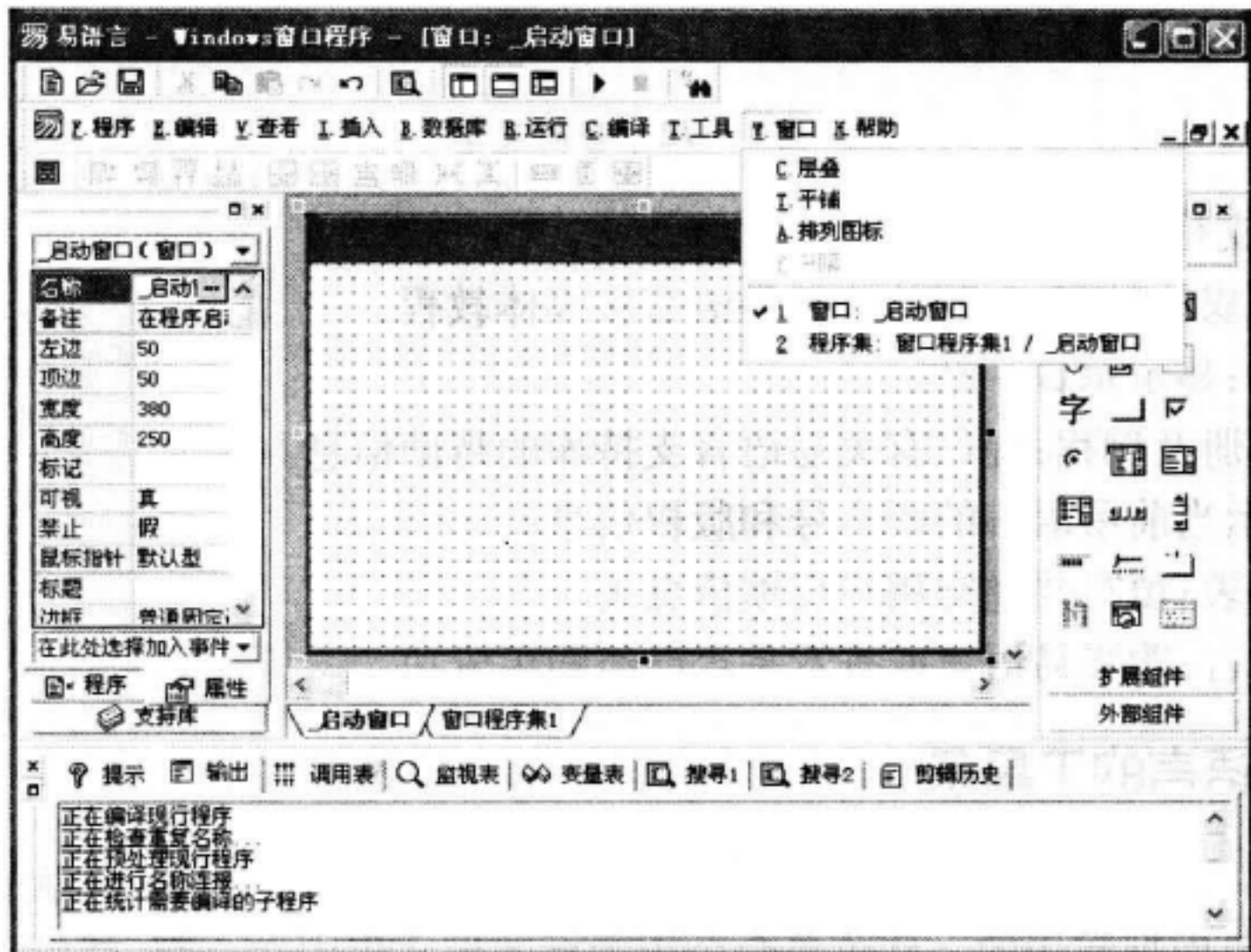


图 1-71 窗口菜单

窗口菜单中各选项的说明如下:

- 层叠:排列窗口成相互重叠。
- 平铺:排列窗口成互不重叠。
- 排列图标:将图标排列在窗口底部。
- 分隔:将活动的窗口分隔成窗格。
- 窗口:已被激活的设计窗口。

10. “帮助”菜单

“帮助”栏中包括切换到“即时帮助”提示子夹、“易语言知识库”等,如图 1-72 所示。

帮助菜单中各选项的说明如下:

- 即时帮助:在状态夹中显示有关当前位置的帮助信息。即时帮助信息内容是在用户进行任何操作的同时,随时按下热键“F1”将有关的支持信息在提示面板中显示出来。
- 易语言知识库:打开并显示易语言知识库。易语言知识库文档基本包含了当前易语言版本中所有帮助信息,并且配有大量的贴图和源代码,为学习易语言提供了很大帮助。易语言

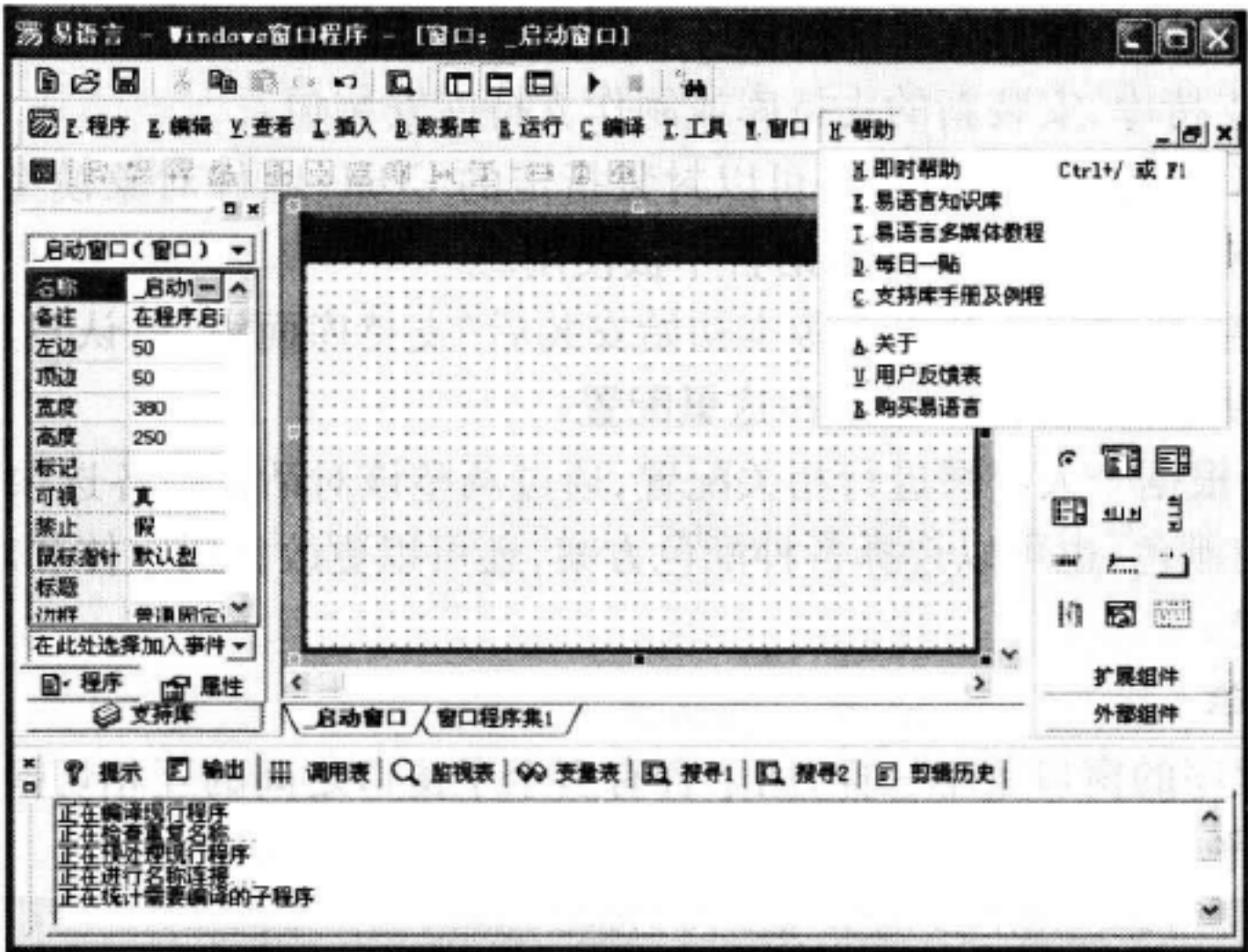


图 1-72 帮助菜单

支持库需要安装支持库文件后,才可使用。

- 易语言多媒体教程:打开并显示易语言多媒体教程。
- 每日一贴:显示每日一贴。
- 支持库手册及例程:提供有关易语言支持库的帮助信息。
- 关于:显示当前易语言的版本号和版权信息。
- 用户反馈表:填写并递交用户反馈信息表。
- 购买易语言:购买易语言或查看是否已经购买成功。

1.3.3 易语言的工具条

工具条,也称为工具栏,如图 1-73 所示。众所周知,工具栏就是菜单条的快捷方式,为了避免多次选择菜单而把常用的一些菜单命令用图标的形式集中在一起,就成了工具条。因此,工具条中的所有命令大多都可以在菜单栏中找到原型。

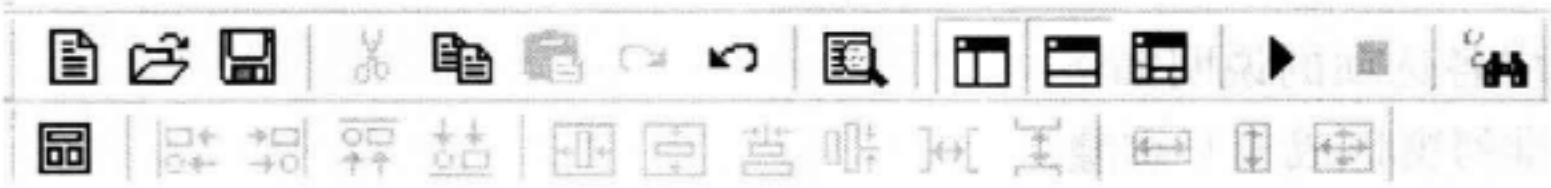


图 1-73 工具栏

在“查看”菜单中选“工具条”,或者在工具条上单击鼠标右键,都会出现如图 1-74 所示的菜单。在菜单中选择指定的工具栏前打“√”号,则该工具栏就会出现在主窗口上;消去“√”号,则该工具栏会隐去。

易语言的工具条有两类:“标准工具条”和“对齐工具条”。

“标准”工具条主要包括最常用的十余个命令,包括程序操作、编辑操作、执行命令、窗口布局等。

“对齐”工具条包括左/右/顶/底/中对齐,分布间距,等高度/宽度等。

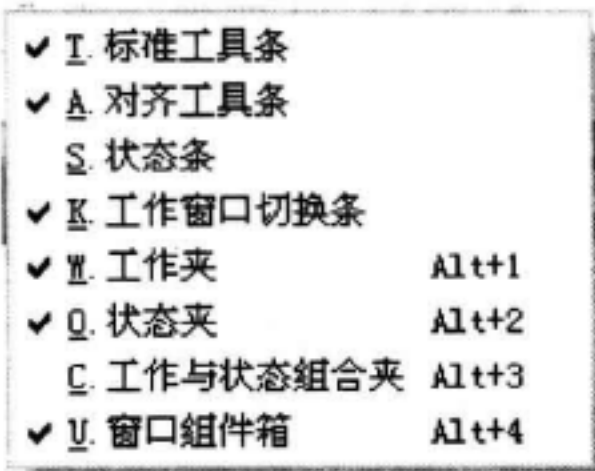


图 1-74 工具条菜单

1.3.4 如何在设计窗口中添加组件

从组件框中选出所需的组件添加在设计窗口中,只需要用鼠标左键在组件框中点击欲添加的组件,使其处于选中状态,然后在设计窗口中按住鼠标左键拖动,拉出一个该组件即可。添加后的组件可以通过拖动鼠标改变其位置和大小,也可以使用方向键来微调组件的位置,还可以按住 Shift 键来微调组件的大小。

1.3.5 如何使用易语言帮助系统

易语言的帮助系统分为“即时帮助信息”和“易语言知识库”。

1. 即时帮助信息

易语言编程环境在用户进行任何操作的同时,会将有关的支持信息在提示面板中显示出来,可以使用以下介绍的方法来查看即时帮助信息:

随时按下“F1”热键使用可随时得到与主题相关的帮助信息。

使用菜单:“帮助”→“即时帮助”可得到与主题相关的帮助。即时帮助信息内容是在用户进行任何操作的同时,将有关的支持信息在提示面板中显示出来。

即时帮助信息可显示系统中各运行支持库内的命令、库定义数据类型、库定义常量等等信息。直接在工作夹内的支持库面板中找到并单击欲查找信息的项目,此时所有的相关信息将会显示在状态夹的提示面板中。

如果欲将这些信息提取出来打印或者以后阅读,可以在相应项目上单击鼠标右键,在弹出菜单中选择“拷贝帮助文本到剪贴板”或者“写帮助文本到文件”,输出与该项目及该项目所有子项目相关的帮助信息,供在计算机上浏览或打印出来阅读。

2. 易语言知识库

易语言的帮助文档已经相对成熟了,包含了当前易语言版本的所有帮助信息,以及大量的贴图和源代码,为学习易语言提供了很大帮助。

打开易语言知识库可以通过点击易语言“帮助”菜单中的“易语言知识库”选项或直接点击易语言工具条中的知识库按钮,如图 1-75 所示。

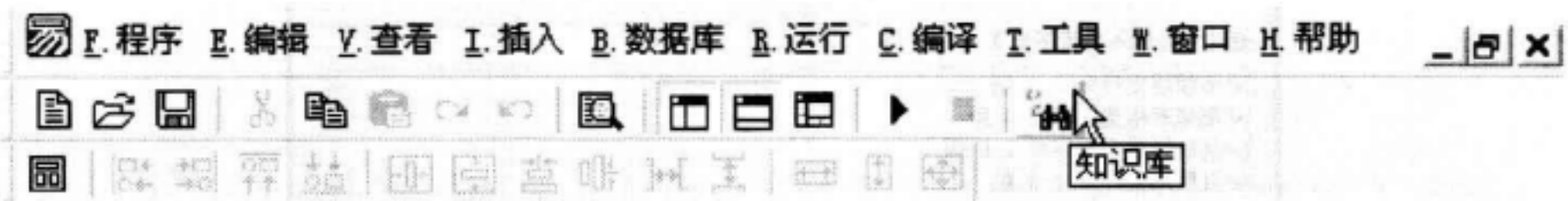


图 1-75 知识库按钮

3. 前层提示信息

易语言中,每输入一个命令代码,将鼠标移动到该命令上,都会出现一个信息提示框,显示该命令的帮助信息,如图 1-76 所示。

子程序名	返回值类型	公开	备注
_按钮1_被单击			

结束 0

《无返回值》 结束 () - 系统核心支持库-流程控制
解释: 本命令结束当前易程序的运行。

1.3.6 如何配置易语言

1. 系统配置

点击菜单“工具”→“系统配置”。可以打开易语言的系统位置对话框,通过调整该对话框中各项属性的参数,可以自定义界面各部位颜色,可以选择各种配色方案,还可以更改代码字

图 1-76 易语言前层提示

体,和对内置输入法等很多方面进行配置,如图 1-77 所示。

2. 程序配置

点击菜单“程序”→“配置”,可以打开程序配置对话框,如图 1-78 所示。

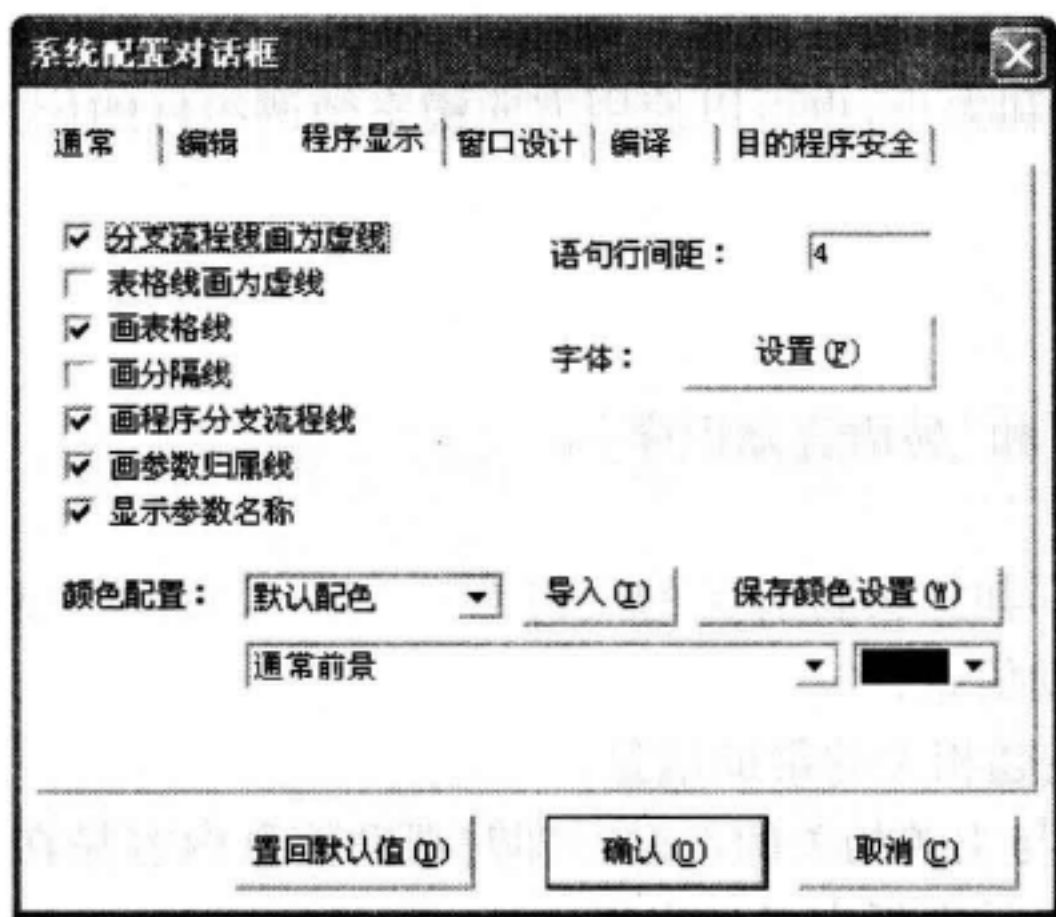


图 1-77 系统配置对话框

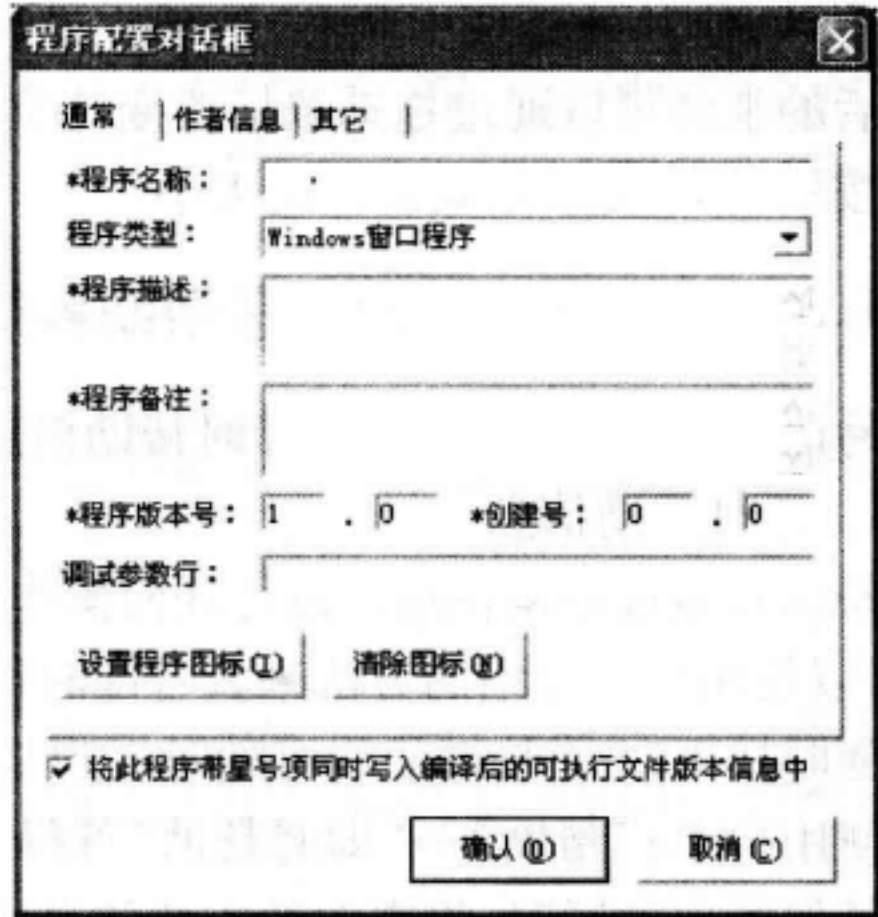


图 1-78 程序配置对话框

该对话框可以将程序名称、程序备注、作者信息等信息保存在生成后的 EXE 文件中,当查看此 EXE 文件的属性时,这些信息会显示出来,并且可以在这里为程序设置图标。

如果每次要生成 EXE 文件,然后运行带参数的 EXE 文件是非常麻烦的,在本对话框中输入“调试参数行”后,当调试运行该程序时,易语言将自动附加该参数。生成 EXE 文件后,此调试参数行内人不会保存至程序中。

3. 支持库配置

易语言 3.8 以上版本初始安装后“支持库面板”只显示系统核心支持库,如果使用到更多的文件库可以通过点击主菜单“工具”→“支持库配置”,可以打开支持库配置对话框,如图 1-79 所示。

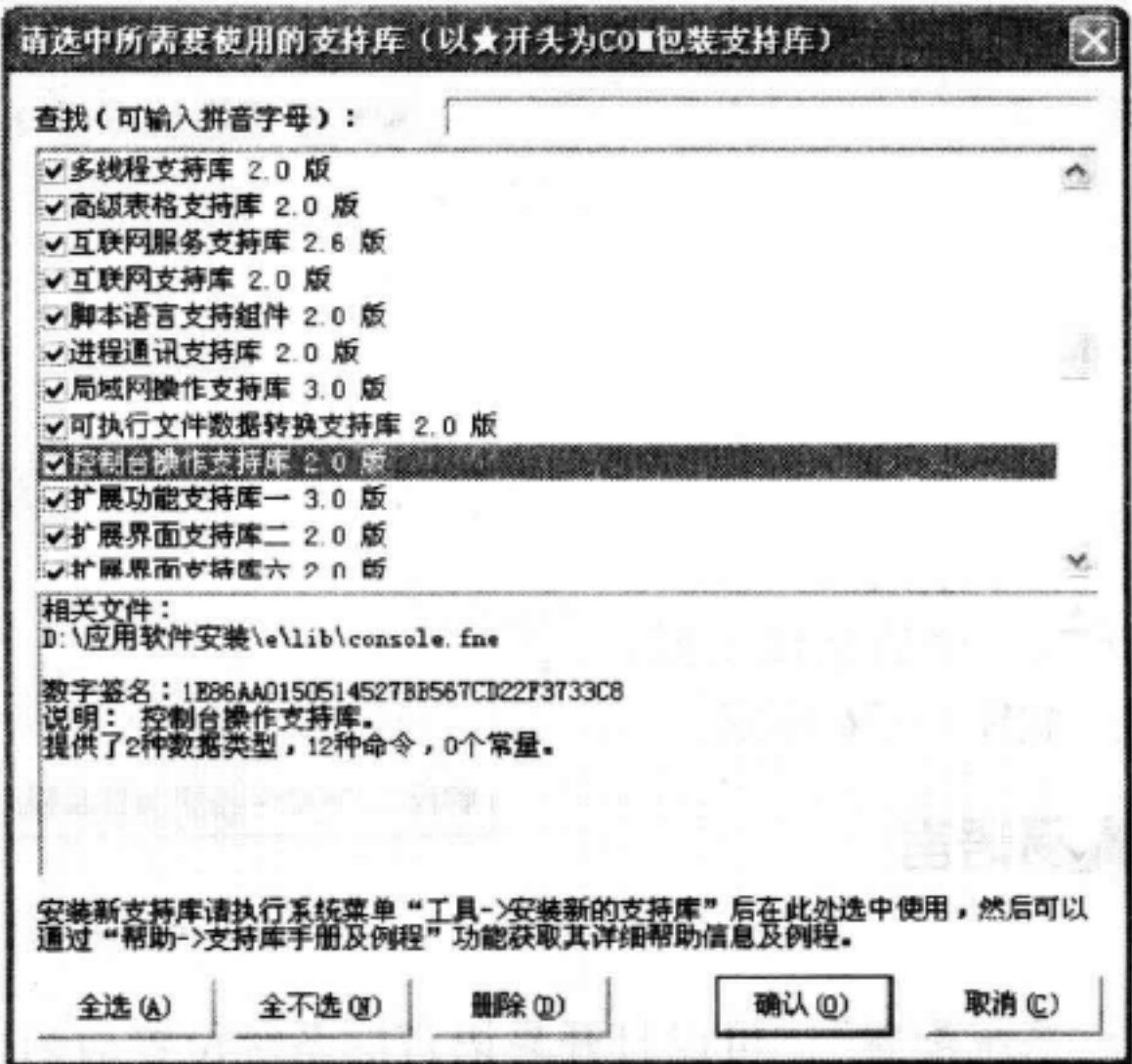


图 1-79 支持库配置对话框

点击支持库列表中任一项,可查看该支持库的基本信息。如果将某支持库名称前的对勾“√”加上,则在易语言的“支持库面板”列表中显示该支持库,并可以在程序中使用其提供的命令、数据类型、变量等。

1.4 易语言代码输入方法

易语言作为一种全中文编程语言,在其输入代码时是具有一些技巧的。在易语言刚运行时,直接点击提示面板,可以看到关于易语言程序输入方法的详细提示信息,如图 1-80 所示。

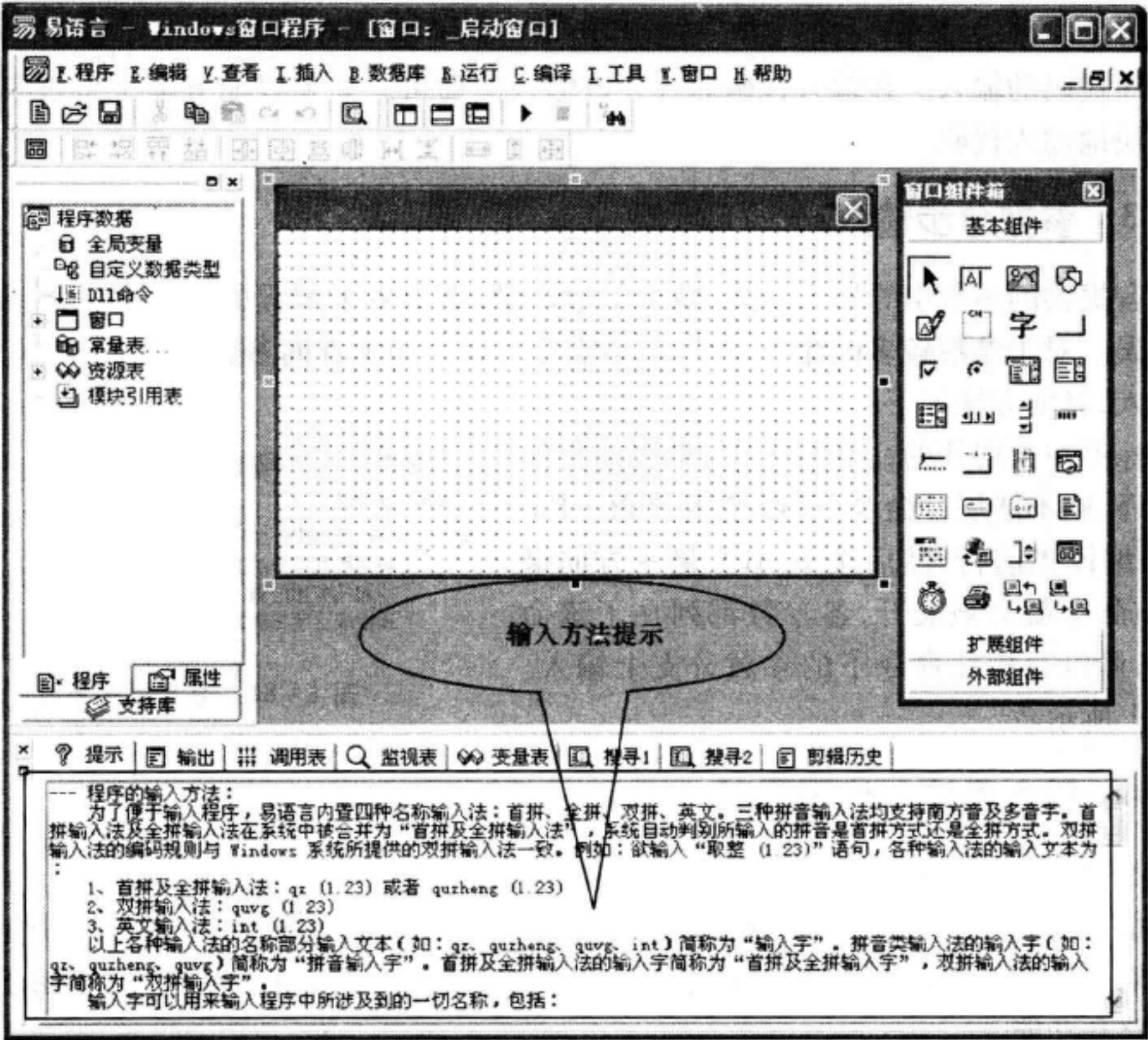


图 1-80 程序输入方法提示

1.4.1 内置输入法

在第 1.1 节易语言概要部分,曾介绍易语言内置有 4 种输入法:首拼、全拼、双拼、英文,其中 3 种拼音输入法均支持南方音及多音字。首拼输入法及全拼输入法在系统中被合并为“首拼及全拼输入法”,系统自动判别所输入的拼音是首拼方式还是全拼方式。双拼输入法的编码规则与 Windows 系统所提供的双拼输入法一致。例如代码:取绝对值(36),各种输入法的输入文本如下:

- 首拼输入法:qjdz(36)
- 全拼输入法:qujueduizhi(36)
- 双拼输入法:qujtdvvi(36)

• 英文输入法:abs(36)

代码中涉及到的汉字,都可以使用这几种内置输入法输入。有一些特殊情况,比如对于没有声母的汉字(如“按”),使用首拼输入法时应取其韵母全部字符。比如“按钮”,用首拼输入法就可以输入成“ann”,其中“an”为“按”的韵母,“n”为“钮”的声母。如果要输入的名称中既有汉字又有英文字母,则其中的英文字母不论大小写都要用大写英文字母输入。例如,要输入“编辑框 x”,使用首拼输入法就要输入“bjkX”。

1.4.2 系统输入法

使用五笔字型、自然码、智能 ABC 等这些由 Windows 提供的系统输入法,在易语言中也可以进行程序代码的输入。在输入代码时打开该输入法即可。如果以前五笔字型很熟练的话,也可以很快地输入代码。

1.4.3 参数分步输入

易语言提供的参数引导输入功能,减少了记忆量,更节省了编程的时间,极大降低了程序录入的错误。对于参数较多的命令,程序员不需要再花时间去查询参数的意义,可以直接将命令展开输入,具体方法如下:

将光标停在欲展开的命令行上,如果当前行没有通过编译,则不能展开命令,可以使用 Shift + Enter 键来预编译当前行,然后按下 ALT 键 + 方向键的右键,该命令就会被展开,各参数都列在了该命令的下面,可以直接在命令下的参数分支上输入。如图 1-81 所示。

子程序名	返回值类型	公开	备注
按钮1_被单击			

取文本中间 (编辑框1.内容, 3, 7)
※欲取其部分的文本: 编辑框1.内容
※起始取出位置: 3
※欲取出字符的数目: 7

图 1-81 分步输入参数

1.4.4 输入备注与代码屏蔽

1. 备注输入

备注是一行或一段代码的提示和说明。编写代码时一定要养成一个良好的习惯,就是给部分代码输入备注信息,这样做既方便了自己日后阅读代码,又方便了其他人更快地理解程序代码的思路和功能。

备注输入的方法是在备注文字前加“'”号,则该符号后的本行文字变为备注,在输入代码时,可以在代码的旁边或代码的下方输入备注。如图 1-82 所示。

2. 屏蔽与批量屏蔽

在任何情况下,如果想屏蔽一行代码,则再该行代码前加“'”号,和置为备注的方法相同,屏蔽后的代码在运行调试时不会被编译,调试程序寻找错误时,该方法会起到很大作用。将代码前的“'”号删除便可以解除屏蔽了。

除此之外,也可以在代码上单击鼠标右键,弹出的菜单中又“屏蔽”和“解除屏蔽”的选项。屏蔽的快捷键是 Ctrl + K 键,可以选中多行代码,然后使用 Ctrl + K 键来屏蔽所选的代码,然

子程序名	返回值类型	公开	备注
比较大小	整数型		

变量名	类型	静态	数组	备注
a	整数型			
b	整数型			

如果 (a > b)
返回 (a)
 如果a的值大,则返回a的值
返回 (b)
 如果b的值大,则返回b的值

图 1-82 输入备注

后使用 Ctrl + M 键来解除代码的屏蔽。

1.4.5 4 种输入语句

易语言常见的程序语句分为 4 种:赋值型语句、非运行语句、方法语句、命令语句。

1. 赋值型语句

用于给某组件属性或变量赋值的语句。一般使用“=”连接被赋值方和赋予的值,并且赋予的值一定要和被赋值的属性或变量的数据类型相同或互相兼容。例如:

2. 给组件属性赋值

标签 1. 标题 = “易语言正式版”

3. 给变量赋值

变量 1 = 29000

4. 非运行语句

非运行语句是在运行过程中不被运行的语句,包括注释型语句和草稿型语句。

5. 命令语句

命令语句也就是执行命令使用的语句。命令是一种程序运行动作指令,易语言的命令由易语言基本支持库和扩展支持库提供,命令的调用格式为:命令名(参数,...),例如:

信息框(“欢迎使用易语言!” ,0,)

其中括号前是命令名,括号中的内容是命令的参数,主要是供命令进行判断、选择或再加工的因素,每个参数用逗号分隔。

6. 方法语句

方法是一个具体对象能够执行的动作,方法语句就是用于执行某对象的方法而使用的一类语句,对象中的方法是通过发送消息实现。方法的使用格式和命令类似,很多方法都有参数表,调用格式为:对象. 方法名(参数 1,参数 2,...)。一般方法都是指组件的方法。例如:

表格. 插入行(第三条记录,3)

第 2 章 易语言基础知识

各种数据存放在磁盘或内存中都有其不同的存放格式,因此就存在不同的数据类型。了解各种数据类型的特性,对编程开发相当重要。

程序运行过程中所存放的临时数据主要通过变量保存,编写程序离不开变量的使用。

程序中经常会进行一些运算,易语言中的运算都要使用运算符进行识别处理,并通过运算表达式来完成运算操作。程序中对各数据之间的关系的描述也要通过运算符。

2.1 易语言数据类型

一个程序内部应包括两个方面的内容:数据的描述、操作步骤。操作步骤即对程序动作的描述。

数据是程序操作的对象,操作的结果会改变数据的内容。就像厨师做菜,做菜前需要选择好烹饪的原材料(即对数据进行描述),然后开始烹饪(即对数据进行操作),最后做好了一道菜(改变了原先数据的状况,得出了计算结果)。

编写程序也一样,程序要对一些数据进行操作,在操作前首先要对被操作的数据进行描述,即定义相关数据类型的变量,然后再用命令或者方法来对该数据进行操作,最后得到操作结果,进一步可将结果显示出来。

易语言的数据类型从数据结构来区分,可分为基本数据类型和复合数据类型。基本数据类型包括:数值型、文本型、逻辑型、日期时间型等;复合数据类型包括支持库自定义数据类型和用户自定义数据类型。

数据类型可以用来描述变量的类型或组件属性的类型等,对于变量和常量将在下一节进行介绍。

1. 系统基本数据类型

(1) 数值型。生活中有很多数据都是按照数字方式进行记录的,它们都能执行加减乘除等运算,但不同的数字也有不同,有整数的、有小数的,因此,为了满足不同的计算,易语言需要提供多种数字类型,具体如表 2-1 所示。

表 2-1 数值型数据

数据名称	数据类型长度
字节型	0~255 个字节
短整数型	-32 768 到 32 767 之间的数值,尺寸为 2 个字节
整数型	-2 147 483 648 到 2 147 483 647 之间的数值,尺寸为 4 个字节
长整数型	-9 223 372 036 854 775 808 到 9 223 372 036 854 775 807 之间的数值,尺寸为 8 个字节
小数型	3.4E+/-38(7 位小数)之间的数值,尺寸为 4 个字节
双精度小数型	1.7E+/-308(15 位小数)之间的数值,尺寸为 8 个字节

数值型数据容纳的数值范围越大,占用的字节也就越多。例如,短整数型的数值 3000 和整数型的数值 3000,都代表了数值 3000,但在系统中占用的空间却不同,即短整数型占 2 个字节,整数型占 4 个字节。所以,在实际应用时要根据需要来选择使用的数据类型,避免存储空间的浪费。例如,存储的数据在 -32768 至 32767 以内,就要采用短整数型;如果使用小数而对精度要求不高,就可以使用小数型而不采用双精度小数型。

当某类数据类型存储的值超出了其所能容纳的范围,就会发生数据溢出错误。例如,当短整数型数据存放大于 32767 的数值时,就会得出错误的结果。所以在选择数据类型时,除了要避免空间的浪费,还要防止数据的“溢出”。

不同的基本数据类型之间是可以任意转换的。不过编程需要注意转换时可能带来的精度丢失。例如,将整数 257 转换为字节后的结果为 1,这是因为 257 超出了字节型数据的最大上限 255,从而产生了“溢出”。

(2) 文本型。除了数字之外,更常见的数据就是按照文本形式存在的,其中有英文、中文,或者数字。在易语言中被视为文本型的数据,用两个引号括起来,例如“张三”、“李四”,等等。

(3) 逻辑型。前面提到的一个人是否结婚只能用“是”或者“否”来表示,与此类似的有,一个窗口是否可见、是否可以被移动等等,在计算机中这类属性统一用真/假来表示。易语言中称之逻辑型数据,其值只可能为“真”或“假”,长度为 2 个字节。“真”和“假”为系统预定义常量,其对应的英文常量名称为“true”和“false”。

(4) 日期时间型。这类数据用作记录日期及时间,长度为 8 个字节。

(5) 字节集。字节集是易语言特有的格式,用作记录一段字节型数据。字节集与字节数据之间可以互相转换,在程序中允许使用字节数组的地方也可以使用字节集,或者相反。字节数组的使用方法,例如用中括号(“[]”)加索引数值引用字节成员,使用数组型数值数据进行赋值等,都可以被字节集所使用。两者之间唯一的不同是字节集可以变长,因此可把字节集看作可变长的字节数组。

2. 库定义数据类型

库定义数据类型由运行支持库提供,用户在程序中可以直接使用,就如同是系统基本数据类型一样。易语言中提供的每一种内部组件都可以作为一种数据类型,例如:图片框、列表框、编辑框、按钮,等等。这些数据类型具有组件的特征,如属性、方法等,都和组件完全相同,在程序中也可以当成一个组件来使用,因此在易语言中也被称为特殊数据类型。

例如,窗口中有 1 个表格组件“表格 1”,程序中可以直接将该表格组件赋值给一个数据类型为“表格”的变量 1,代码如下:

```
变量 1 = 表格 1
```

这样的话,变量 1 就具有了表格的属性、方法。可以通过以下语句来为该变量设置属性:

```
变量 1. 宽度 = 480
```

```
变量 1. 高度 = 400
```

3. 用户自定义数据类型

除了使用易语言提供的数据类型以外,还可以根据需要自定义新的数据类型。自定义数据类型时需要设置数据类型的名称及其成员。数据类型成员各属性的设置方法等同于容器设置时相应属性的设置方法。

例如,定义一个三维坐标点,会有 X、Y、Z 一组的指标数据,它们出现并且有一定的联系。

具体定义方法如下：

步骤 1：在程序面板双击“自定义数据类型”，如图 2-1 所示。程序会自动切换到“自定义数据类型”的界面。

步骤 2：在自定义数据类型界面按下 Ctrl + N 键，新建一个数据类型，如图 2-2 所示。或者直接单击“插入”菜单→“数据类型”来设置自定义数据类型，能达到同样的设计效果。

步骤 3：将该数据类型定义为“三维坐标”。由于三维坐标的位置取决于 X、Y、Z 三轴的坐标，所以，在“成员名”上按 3 次回车，加入 3 个成员。将 3 个成员名分别定义为“X 位置”、“Y 位置”、“Z 位置”，并将 3 个成员的数据类型都定义为整数型，如图 2-3 所示。

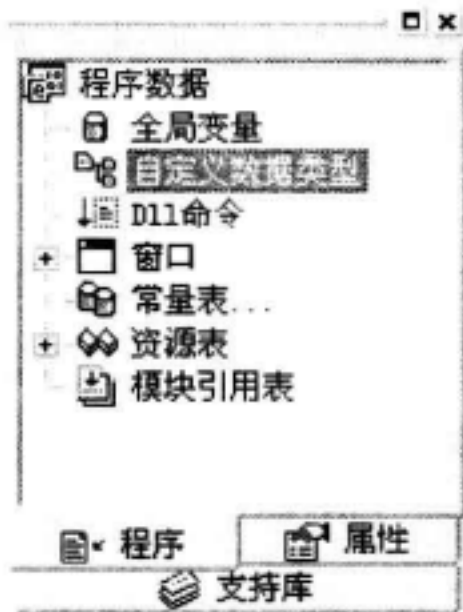


图 2-1 自定义数据类型

数据类型名	公开	备注		
数据类型1				
成员名	类型	传址	数组	备注

图 2-2 新建数据类型

数据类型名	公开	备 注		
三维坐标				
成员名	类 型	传址	数组	备 注
X位置	整数型			
Y位置	整数型			
Z位置	整数型			

图 2-3 定义“三维坐标”数据类型

这样在程序中就可以使用“三维坐标”这个自定义的数据类型来保存这一组整数型的数据了。

2.2 变量与常量

2.2.1 变量

变量没有固定值，就像二元一次方程式中的 X、Y 一样。但具有一个名字，可以存储可变的数据，在内存中占据一定的存储单元，单元中存储的值就是该变量的值。可以把变量看成一个容纳东西的容器，只是这个容器容纳的是各种可变的数据。就像买东西时用的购物袋，购物袋里的东西应该是能变化的，可以是吃的，也可以是喝的、用的，可以放进去，也可以拿出来。变量也一样，可以提前变量中的数据，也可以改变变量中的数据。

变量在程序中的应用非常广泛。不但可以用来存放各种类型的数据，而且还可作为命令的参数使用，例如用于“计次循环首()”命令，命令的第二个参数变量会记录循环的次数。

变量名可以任意定义，根据需要给变量定义一个实际意义明显的名字，例如：“存放水表现在数据的变量”，或者直接定义变量名为“水表现数”。这样定义变量名，不仅有助于日后查看代码时很快了解该变量的作用，而且也方便了与他人交流代码。尤其是在变量使用频率很高的时候，其定义的变量名显得尤为重要。养成一个良好的定义名称的习惯，对以后的编程将有很大的帮助。需要注意的是，定义的变量名的首字符不可以是数字，同时变量名中不能出现“_”之外的其他符号和标点。这些规则都是为了防止变量名和程序中的数值或符号重复，造成程序的混乱。

变量有多种类型，不同种类的变量其特性也有所不同。从变量的作用范围来区分，可以将变量分为“局部变量”、“全局变量”和“程序集变量”。

(1) 局部变量。局部变量,顾名思义,只能在其所在的子程序中才能被调用,其他子程序都无法调用。因为子程序被调用的时候,这种变量才占用系统的内存,当子程序执行结束后,变量所占空间被系统收回,因此局部变量非常节省系统内存。

(2) 全局变量。在程序运行后,所有程序集内子程序都可以使用的变量称为全局变量。全局变量是覆盖范围最广的变量。这种变量在程序运行后即占用内存空间,在程序运行结束后才从内存中清除,所有会长时间占用系统资源,建议根据程序的实际情况适当地使用全局变量。

(3) 程序集变量。一般情况下,程序集变量仅在本程序集中被调用。若在其他窗口程序集中调用,则需要在变量名前加程序集所对应的“窗口名称”前缀,例如:

信息框(_启动窗口.变量1,0,)

程序集变量所在的程序集中的所有子程序,都可以自由访问程序集变量,多个子程序都需要访问的数据,可以使用程序集变量来存储。

在选择使用变量的类型时,尽量选择符合该变量适用范围的变量类型,以节省系统内存。

若从变量的属性来区分,变量还可分为“静态变量”和“数组变量”

(1) 静态变量。精致存在的局部变量就是静态变量。当所处子程序退出时,静态局部变量能够保留住现行内容以供下次继续使用;而非静态的变量就不能,下次进入子程序时它将被重新初始化。如果局部变量不设置“静态”属性,子程序执行完毕后,将清空该子程序中的所有非静态局部变量;如果局部变量设置了“静态”属性,当子程序执行完毕后也不会被清空,当子程序再次被调用时,静态变量的值仍保持上次被调用时的状态。

(2) 数组变量。用来存放一组数据的变量,即数组变量。数组变量中的每个成员都拥有独立的存储单元,可以单独调用和赋值。其实数组变量可以看作是多个非数组变量组成的。数组变量又分为“单维数组变量”和“多维数组变量”。

定义变量的方法有很多,比如用 Ctrl + L 键新建“局部变量”,并指定变量的名称和数据类型;可以用 Ctrl + G 键新建一个“全局变量”。除此之外,还可以在易语言的“插入”菜单中选择“全局变量”或“局部变量”来插入全局变量和局部变量。如图 2-4 所示。

在“窗口程序集名”上按回车键,可以插入新的“程序集变量”,如图 2-5 所示。

变量必须先定义才能够被使用,如果代码中引用了没有定义的变量,则该行代码不会通过编译,状态夹中会有相应的错误提示,如图 2-6 所示。在编写代码时,如果涉及的变量比较多,可以在代码中使用一些未定义的变量,但一定要记住代码输入完后补建这些变量,否则无法通过编译。

通过对变量赋值来实现变量存储数据的功能。给变量赋值的时候要注意变量的数据类型,要符合各数据类型的赋值规则,还有几点需要注意。

(1) 给数值型数据赋值时,数据会自动转换类型。任意数值类型的数据可以被写入到其他任意数值类型的变量中,系统将自动进行转换。比如,将一个短整数写入到整数型变量中,

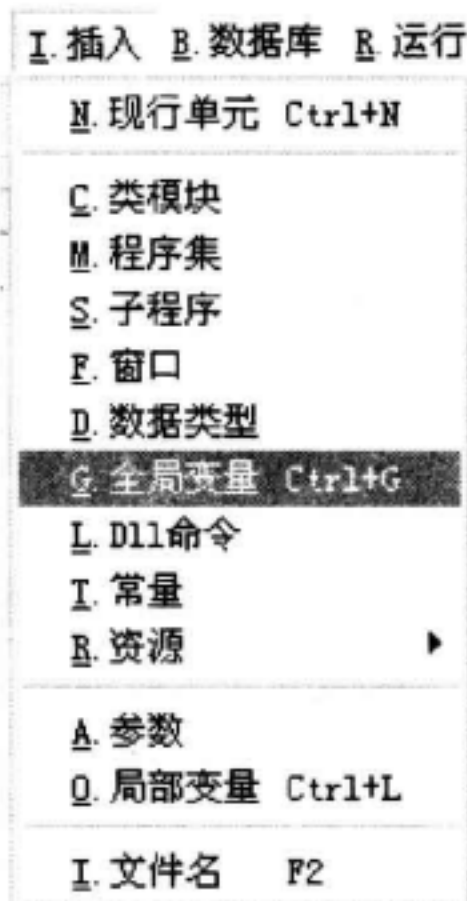


图 2-4 新增全局变量

窗口程序集名	保留	保留	备注
窗口程序集15			
变量名	类型	数组	备注
程序集变量			

图 2-5 新增程序集变量

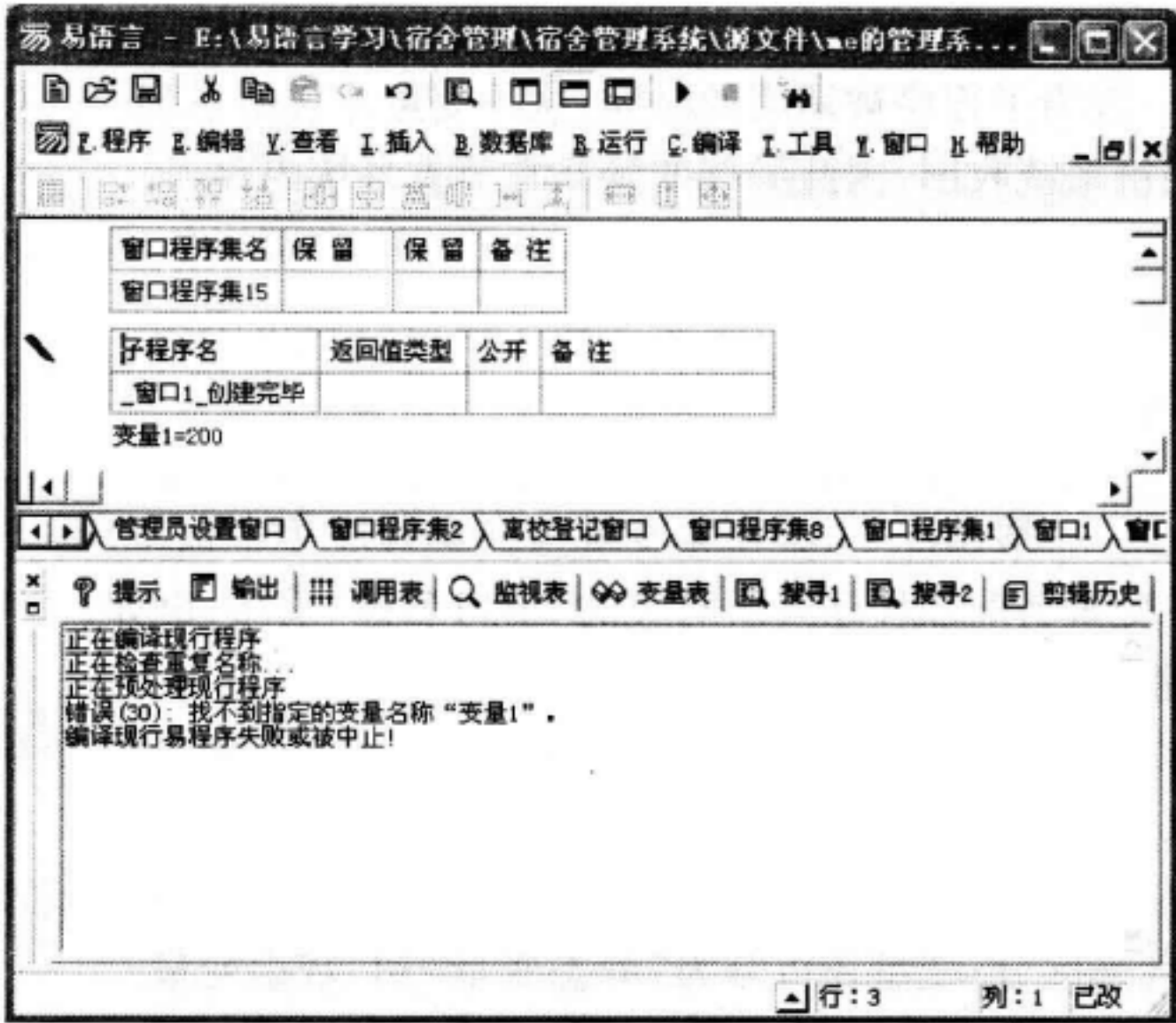


图 2-6 使用未定义变量的错误提示

将一个整数写入到小数型变量中等的,此时要注意防止出现上一节提到的“溢出”问题。另外要注意,若将小数型变量写入到整数型变量时,会丢失小数点后的内容,因此最好转换类型是一一对应。

(2) 使用“连续赋值”命令给多个变量同时赋值。连续赋值命令由 2 个参数,第一个参数是用作赋予的值和资源,第二个参数是被赋值的变量或变量数组,第二个参数可重复添加,即可以添加多个被赋值的变量。例如有如下语句:

连续赋值(100,变量 1,变量 2,变量 3)

这行代码实现的功能是给变量 1、变量 2 和变量 3 同时赋值 100。这条代码相当于以下 3 行代码:

变量 1 = 100

变量 2 = 100

变量 3 = 100

程序运行后,变量在没有被赋予新的数值之前,会有个初始数据。这个初始数据被称为变量的初始值。不同数据类型的变量,其初始值也有所不同,具体参照表 2-2。

表 2-2 变量的初始值

变 量 类 型	变量初始值	初始值在代码中的表示方法
数值型	0	0
逻辑型	假	假
日期时间型	1899 年 12 月 30 日	[1899 年 12 月 30 日]
文本型	空文本	“ ”
字节集型	空字节集	{ }

2.2.2 常量

常量的值是固定的,不论在什么情况下都不会发生改变。例如圆周率 π ,就可称为常量。易语言中规定了一些常量,例如“#蓝色”代表了数值 16711680、“#F 键”代表了数值 70,所以在易语言程序中使用“#蓝色”,实际上就是调用了“16711680”这个颜色值。

易语言中的核心支持库定义了许多常量,这些常量可以直接用#常量名即可调用,有数值型常量,如颜色值:#蓝色、#绿色;有文本型的常量,如#引号等。易语言的扩展支持库也有许多常量的定义,并且新增的支持库中,有时也会增加新的常量。

在易语言的支持库面板可以对各支持库中定义的常量进行查询。在被展开的支持库项目的最后一项找到“常量”选项,单击“常量”,就可以展开该支持库中的常量列表。单击一个常量后,按下 F1 键,在节目最下面的提示框中可以看到有关该常量的帮助信息,并能查到该常量的值,如图 2-7 所示。如果安装了新的支持库,可以使用同样的方法来查看新支持库中引用了哪些常量,从而方便程序的编写。

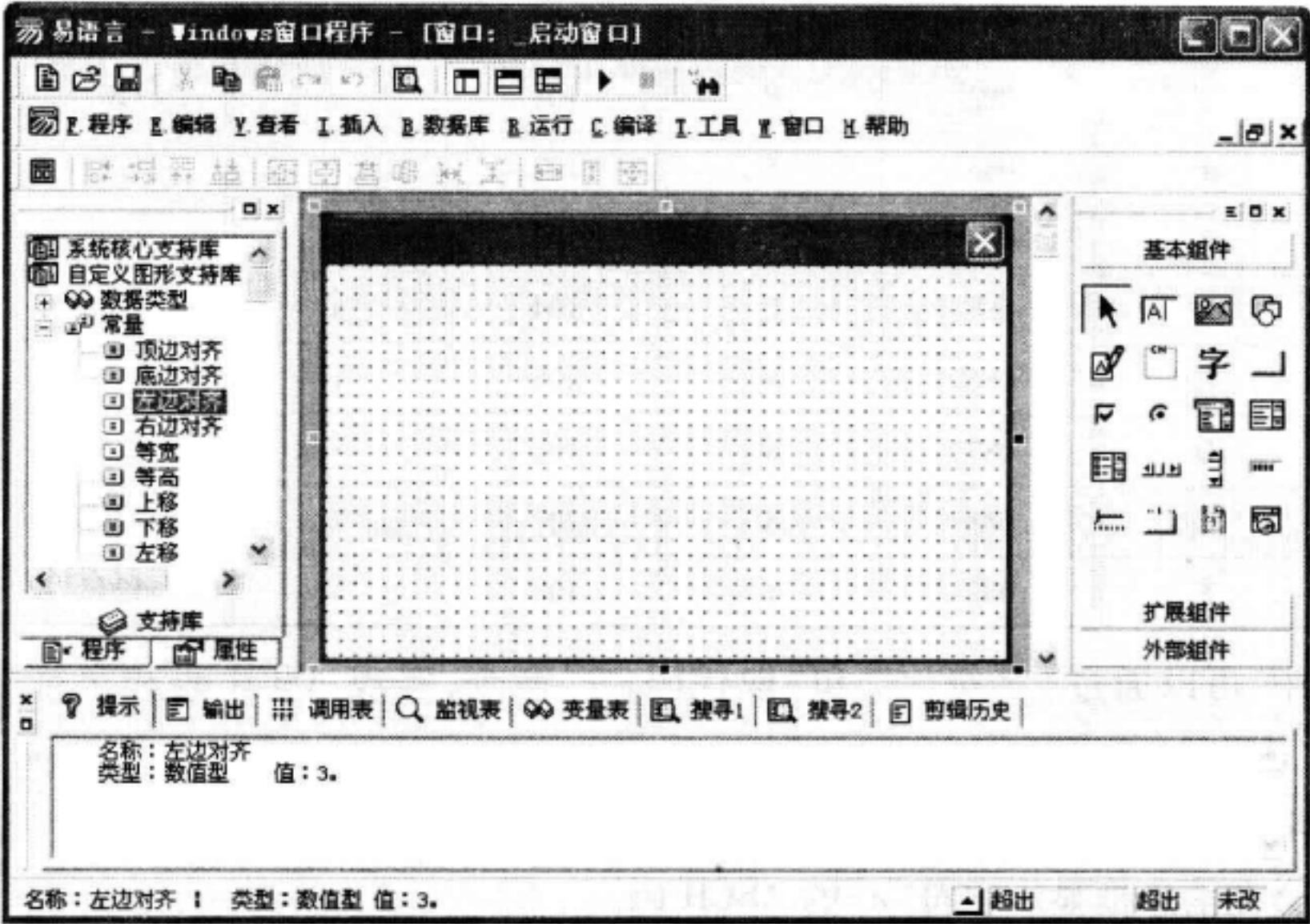


图 2-7 支持库中的常量查询

与其他高级语言一样,易语言中也可以实现 ASCII 码和字符之间的转换。ASCII 码是计算机与因特网中最普遍的文字档案格式,是各种计算机通用的一种常量。例如字符 a 的 ASCII 码是 97、字符 B 的 ASCII 码是 66 等,如表 2-3 所示。

表 2-3 常用 ASCII 表

ASCII 值	对应字符	ASCII 值	对应字符	ASCII 值	对应字符	ASCII 值	对应字符
032	(space)	037	%	042	*	047	/
033	!	038	&	043	+	048	0
034	"	039	'	044	,	049	1
035	#	040	(045	-	050	2
036	\$	041)	046	.	051	3

(续)

ASCII 值	对应字符	ASCII 值	对应字符	ASCII 值	对应字符	ASCII 值	对应字符
052	4	071	G	090	Z	109	m
053	5	072	H	091	[110	n
054	6	073	I	092	\	111	o
055	7	074	J	093]	112	p
056	8	075	K	094	^	113	q
057	9	076	L	095	_	114	r
058	:	077	M	096	`	115	s
059	;	078	N	097	a	116	t
060	<	079	O	098	b	117	u
061	=	080	P	099	c	118	v
062	>	081	Q	100	d	119	w
063	?	082	R	101	e	120	x
064	@	083	S	102	f	121	y
065	A	084	T	103	g	122	z
066	B	085	U	104	h	123	{
067	C	086	V	105	i	124	
068	D	087	W	106	j	125	}
069	E	088	X	107	k	126	~
070	F	089	Y	108	l	127	□

易语言中,可以通过“字符()”和“取代码()”命令,实现 ASCII 码和字符之间的转换。
例如:

信息框(取代码(“a”,1),0)

运行后会在信息框显示字符“a”的 ASCII 码。

1. 常用常量的使用方法

(1) 颜色值常量的使用。有颜色属性的组件,在颜色属性上都有一个颜色选择器,用来直接改变颜色,颜色选择器上可直接选择颜色的颜色值都作为常量提供,在调用的时候直接输入“#颜色名”即可,例如:

编辑框 1. 背景颜色 = #桃红

(2) “#换行符”的使用。一段文本尾部加入了一个“#换行符”,接在换行符后面的文本将另起一行,相当于在记事本上输入的回车键。如果想让编辑框显示一段文本并自动换行,就需要使用换行符,将“#换行符”加到欲换行文本的前面即可。

例如:

编辑框 1. 是否允许多行 = 真

编辑框 1. 内容 = “易语言”+#换行符 + “编程可视化”

(3) “#引号”、“#左引号”、“#右引号”。为了避免与代码中表示文本数据的引号相冲突,

程序中将文本的引号作为了一个文本常量,如果要想让编辑框显示出一个引号,就要使用“#引号”常量;若要显示中文标点中的引号,就要使用常量“#左引号”、“#右引号”。

例如:

编辑框 1. 内容 = #左引号 + “我爱易语言!” + #右引号

(4) 键代码的使用。易语言将标准的 101 键盘上所有键的键代码都作为了核心支持库定义的常量,在程序中使用只需要输入“#”+要调用的键名,如键盘上的 F11 的键代码,在易语言中用常量表示为:#F11 键。例如,在向编辑框中输入内容的时候,想屏蔽掉某个键,就可以在编辑框的“按下某键”事件子程序中输入代码,如图 2-8 所示。

子程序名	返回值类型	公开	备注		
_编辑框1_按下某键	逻辑型				
参数名	类型	参考	可空	数组	备注
键代码	整数型				
功能键状态	整数型				

如果其 (键代码 = #N键)
返回 0

图 2-8 键代码的使用

程序运行后,在编辑框中将无法通过键盘输入“N”,但复制、粘贴的方法仍然有效。

(5) 用常量填写参数。很多命令参数填入的都是常量,如:“时间到文本”命令,此命令将制定时间转换为文本并返回。第一个参数为“欲转换到文本的时间”,而第二个参数值可以为以下常量:①#全部转换;②#日期部分;③#时间部分。在填写第二个参数时,可以填写数字,也可以直接填写常量名。

例如:

时间到文本([2004 年 3 月 16 日 5 时 11 分 11 秒],#日期部分)

2. 枚举常量及使用方法

枚举常量是一种使用非常方便的常量类型,它本身是一个常量的集合,将多个常量以成员的形式,存放在一个常量中,使用的格式为:

#枚举常量名. 成员名

枚举常量只是一种常量的表现形式,是由易语言支持库定义的常量集合,调用方法和普通常量相同,但需要注意的是,枚举常量只能由用户来调用,不能自定义。

3. 自定义常量及使用方法

易语言中除了各支持库定义的常量以外,还可以自定义常量,根据需要定义一个新的常量及其代表的数值。编程过程中使用一些自定义常量可以增加程序的灵活性,当程序总多个地方调用了某个自定义常量时,如果改变这个自定义常量的数值,那么多有调用该常量的地方都将自动调整为改变后的值,大大节省了程序修改的时间。

自定义常量可以任意定义常量的名称,然后再“常量值”上输入该常量的值。自定义常量的使用方法和非自定义常量的使用方法相同:

“#”+ 自定义常量的名称

(1) 【范例 2-1】新建一个易程序,在“_启动窗口”中添加一个编辑框组件、1 个标签组件和 1 个按钮组件,单击按钮组件,在标签和编辑框中同时显示出自定义常量的值。具体参考例程 2-1。

说明:单击“插入”→“常量”,自定义一个常量,常量名称为“显示内容”,常量值定义为“我爱易语言”。如果定义数值型的常量,直接在“常量值”上输入数值即可;如果定义文本型的常量,则要在欲定义的文本两端加上双引号,如图 2-9 所示。

(2) 启动窗口程序集如图 2-10 所示。

常量名称	常量值	公开	备注
显示内容	“我爱易语言”		

图 2-9 定义一个文本型常量

窗口程序集名	保留	保留	备注
窗口程序集1			

子程序名	返回值类型	公开	备注
_启动窗口_创建完毕			

子程序名	返回值类型	公开	备注
_按钮1_被单击			

标签1.标题 = #显示内容

编辑框1.内容 = #显示内容

图 2-10 “自定义常量”代码示例

(3) 【运行结果】运行例程 2-1, 观测单击按钮前后标签和编辑框的变化, 如图 2-11 所示。

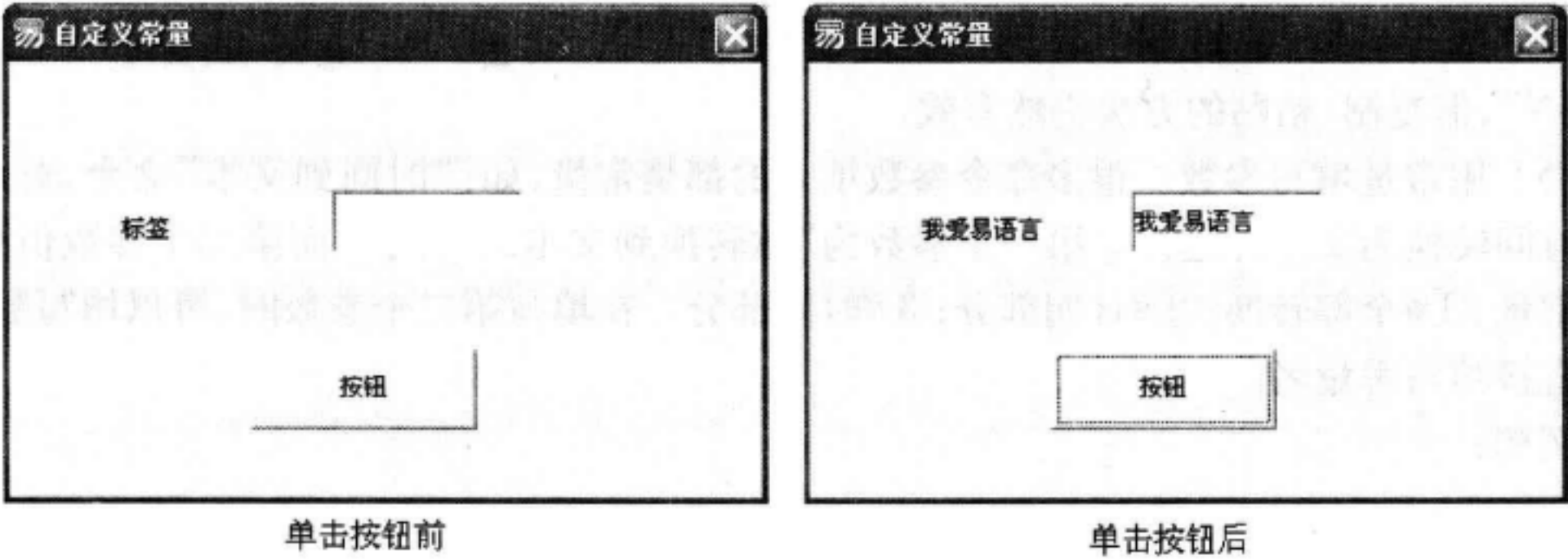


图 2-11 “自定义常量”运行结果示例

(4) 【范例解析】按 F5 键运行程序, 此时编辑框中没有任何内容, 标签的标题属性为默认值“标签”, 单击按钮组件, 标签的标题变成自定义常量的值——“我爱易语言”, 编辑框中显示的内容也为自定义常量的值——“我爱易语言”。

2.3 易语言运算符及表达式

2.3.1 易语言运算符

编写程序代码时, 除了大量的使用命令或对组件的属性、方法进行操作, 运算符的使用也相当重要。程序中所有涉及的算术运算或关系比较运算等操作, 都需要使用运算符。

易语言中提供了丰富的运算符, 参见表 2-4。例如赋值时使用的“=”号, 就是赋值运算符; 比较大小的时候使用的“>”和“<”号, 称为关系运算符等。

表 2-4 易语言运算符


运算符类别	运算符	运算符含义	代码中显示
算术运算符	+	加法运算, 将加号两边的数相加	+
	-	减法运算, 将减号两边的数相减; 负号	-
	*	乘法运算, 将乘号两边的数相乘	×
	/	除法运算, 将除号两边的数相除	÷

(续)

运算符类别	运算符	运算符含义	代码中显示
算术运算符	\	整除运算,将整除运算符两边的数整除	\
	%	求余数运算	%
关系运算符	>	判断是否大于	>
	<	判断是否小于	<
	=或==	判断是否等于	=
	>=	判断是否大于等于	≥
	<=	判断是否小于等于	≤
	<>或!=	判断是否不等于	≠
	?=	判断是否约等于	≈
逻辑运算符	&&或qie	逻辑与运算符,可以连接几个必须同时满足的条件	且
	或huo	逻辑或运算符,可以连接几个可选条件	或
赋值运算符	=	将等号右面的值赋给等号左面的对象	=

易语言中的运算符与其他高级语言运算符一样,都有其优先级别,在程序运行的时候会按照符号的优先级别,从高到低依次运行。易语言常用运算符的优先级参见表2-5。

表2-5 易语言常用运算符的优先级

运 算 符	优先级
()(小括号)	最高
*(乘)/(除)	
\(整除)	
%(求余数)	
+(加)-(减)	
<(小于) <=(小于等于) >=(大于等于) ==(等于) !=(不等于) ?=(约等于)	
&&(逻辑与)	
(逻辑或)	
=(赋值)	

2.3.2 算术运算符和算术表达式

算术运算符在程序中表示方法如下。

- + ,加法运算。例如:3 + 2。
- ,减法也运算或负值运算。例如:16 - 4、- 28。
- × ,乘法运算。例如:4 × 6。
- / ,除法运算。例如:35/4。
- \ ,整除运算。例如:17\3,运算后结果仅保留整数部分,小数部分将被舍去。

% ,余数运算,或者输入“求余数”,第一个参数填被除数,第二个参数填除数,第二个参数可以重复添加。例如:2310% 100、4738% 120% 100。

用算术符号和括号将运算对象连接起来,符合易语言语法规则的式子,称为易语言算术表达式。下面就是一个合法的易语言算术表达式:

变量1 = ((8 × 23 + 45/9) - 11) \ 6

表达式中运算的先后取决于运算符的优先级别,优先级高的运算先计算,反之,优先级低的运算后计算。算术表达式计算的结果可以被程序调用。

(1) 【范例2-2】新建一个易程序,在“_启动窗口”中添加一个标签组件、一个编辑框组件和一个按钮组件,单击按钮组件,将上面表达式的结果通过编辑框显示出来。具体参考例程2-2。

(2) 启动窗口程序集如图2-12所示。

(3) 【运行结果】运行例程2-2,观测编辑框显示的内容如图2-13所示。

窗口程序集名	保留	保留	备注
启动窗口程序集			

子程序名	返回值类型	公开	备注
_启动窗口_创建完毕			

子程序名	返回值类型	公开	备注
_显示结果按钮_被单击			

变量名	类型	静态	数组	备注
变量1	双精度小数型			

变量1 = (8 × 23 + 45 ÷ 9 - 11) \ 6
编辑框1.内容 = 到文本(变量1)

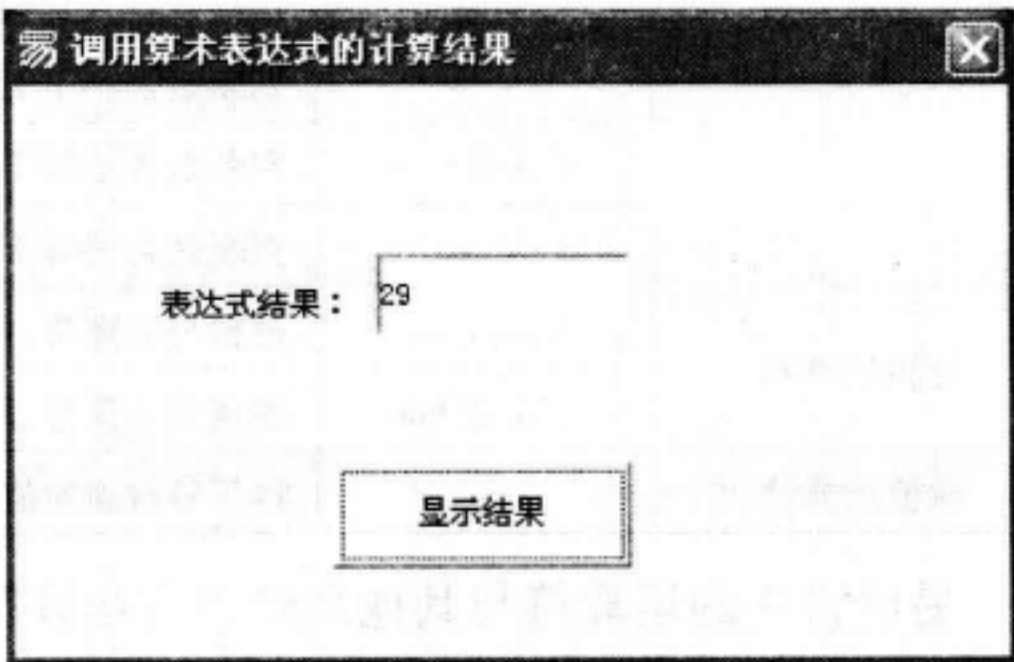


图2-12 “调用算术表达式的计算结果”代码示例

图2-13 “调用算术表达式的计算结果”运行结果示例

(4) 【范例解析】按F5键运行程序,此时编辑框中没有任何内容,单击按钮组件,表达式的计算结果赋值给变量1,并通过编辑框显示。

2.3.3 赋值运算符和赋值表达式

“=”是赋值运算符,在程序中给变量赋值或用代码改变组件的属性,大部分都是使用赋值运算符实现的,将等号右面的值赋给等号左面的赋值对象。例如:

变量1 = 36

编辑框1.高度 = 220

一个正确的赋值表达式,一定要保证欲赋的值和被赋值的对象之间的数据类型要相同,不同的数据类型要转换成相同的数据类型后再赋值。

赋值运算符“=”和关系运算符“=”使用的是相同的符号,但二者的含义却不同。

赋值运算符“=”用于赋值,将“=”右边的值(或变量)赋值给“=”左边的变量(或组件属性、数组成员、自定义数据类型成员);而关系运算符“=”是用于比较符号两边的值是否相等,如果相等返回“真”,不想等则返回“假”。

(1) 【范例2-3】新建一个易程序,在“_启动窗口”中添加一个标签组件、两个编辑框组件和一个按钮组件,比较两个编辑框中输入数值的大小,通过标签的标题属性显示比较的结果。具体参考例程2-3。

(2) 启动窗口程序集如图2-14所示。

子程序名	返回值类型	公开	备注
_按钮1_被单击			

变量名	类型	静态	数组	备注
变量1	整数型			
变量2	整数型			

变量1 = 到数值(编辑框1.内容)
变量2 = 到数值(编辑框2.内容)
如果(变量1 = 变量2)
 标签1.标题 = “相等”
否则
 标签1.标题 = “不相等”

图2-14 “赋值运算符和关系运算符的区别”代码示例

(3) 【运行结果】运行例程 2-3, 观测输入两个数值后比较结果的显示, 如图 2-15 所示。

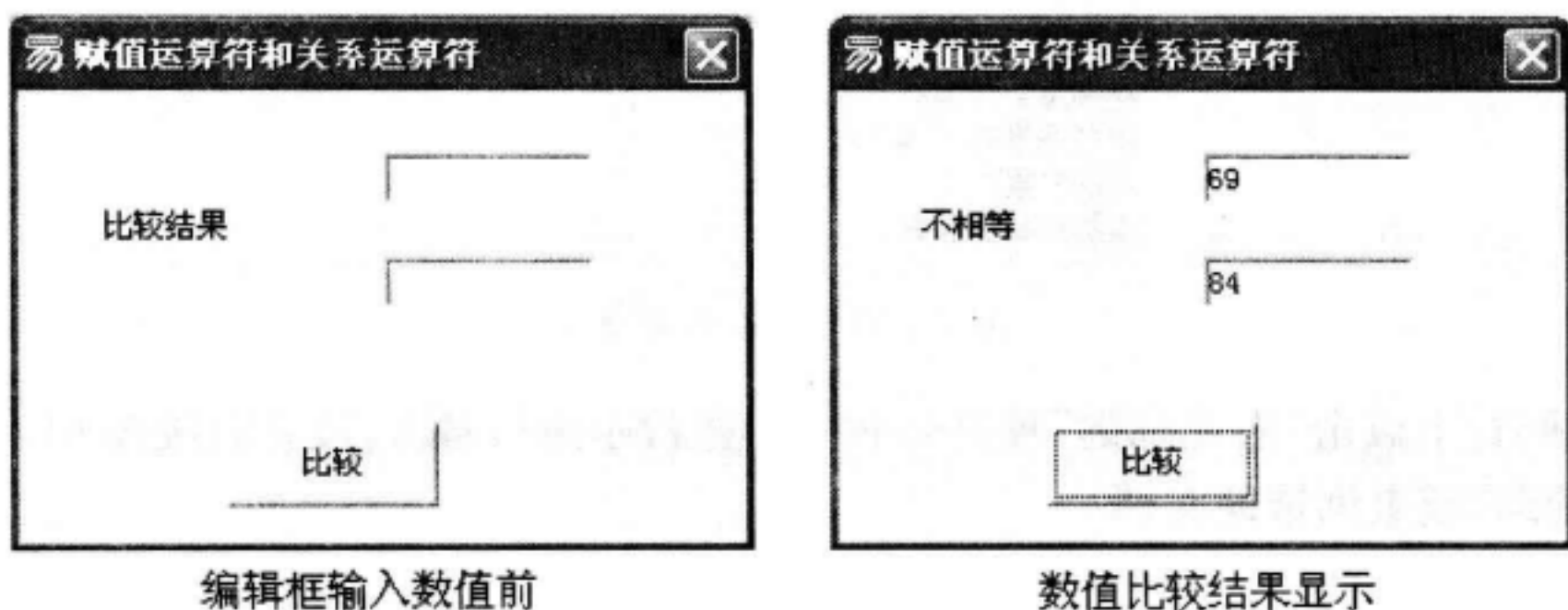


图 2-15 “赋值运算符和关系运算符的区别”运行结果示例

(4) 【范例解析】按 F5 键运行程序, 在两个编辑框中分别输入两个数, 单击比较按钮, 标签的标题显示其比较的结果。上述代码中, 条件语句“如果”中的“变量 1 = 变量 2”, 是用关系运算符“=”进行比较, 如果相等返回“真”, 不相等会返回“假”。如果返回“真”将会执行语句: 标签 1. 标题 = “相等”; 如果返回“假”将会执行语句: 标签 1. 标题 = “不相等”。这两行给标签标题属性赋值的代码中, 使用的就是赋值运算符“=”。

2.4 易语言命令介绍

命令是程序的基本组成部分, 程序是由各种命令组合而成的, 不同的命令完成不同的工作。易语言中提供了大量的命令, 用户可以使用这些命令来实现预想的运行效果。一个程序可以实现一种或多种功能, 而这些功能的实现都离不开程序内部调用的大量命令。

2.4.1 命令的格式

易语言中的命令格式如下:

命令名称(参数,...)

大部分的命令都需要填写参数, 参数用括号括起来, 并用逗号进行分割。部分命令不需要参数, 但是括号不能省略, 例如“结束()”命令。各种命令所要求的个数以及数据类型各有不同, 由其语法决定。例如“到文本()”命令, 该命令只需一个参数, 参数内容为需要转换成文本的数据。有些命令的参数很多, 如“子文本替换()”命令的格式如下:

子文本替换(欲被替换的文本, 欲被替换的子文本, [用作替换的子文本], [进行替换的起始位置], [替换进行的次数], 是否区分大小写)

2.4.2 即时帮助和帮助文档

既然命令有这么多的参数, 记忆起来会比较困难。为了便于用户编写易语言程序代码, 当编写命令时, 按下 ALT + 右键(方向键)可自动展开该命令的参数, 如图 2-16 所示。

易语言核心支持库中已经提供了 600 多个基本命令, 各扩展支持库中也包含了很多命令。为了方便用户使用这些命令, 易语言提供了即时帮助功能和内容丰富的帮助文档。用户不需要将所有的命令语法和参数含义都背下来。在实际的开发工作中, 可使用命令分步输入法

超级列表框1. 插入列 (2, “ 货品名称”, 100, #左对齐, ,)

- ※插入位置: 2
- ※标题: “ 货品名称”
- ※列宽: 100
- ※对齐方式: #左对齐
- ※图片索引:
- ※图片是否居右:

图 2-16 命令参数提示

(在当前代码行上点击“右光标键”展开),根据参数提示进行输入,或者直接按 F1 键查看易语言的即时帮助,或查询帮助文档。

在易语言的帮助系统中,所有的命令都有明确的分类。例如:想对一段文本进行操作,就要查找“文本操作”分类中的相关命令;如果想对一个文件进行操作,就要查找“文件读写”分类中的相关命令。

在易语言支持库面板中,双击展开任意一个支持库名称,可以查到该支持库所有的命令分类;双击展开其中任意一个分类名称,可以看到属于该分类的所有命令;点击任意一个命令名称,就可以在状态夹的提示面板中查看到该命令的即时帮助信息。如图 2-17 所示。

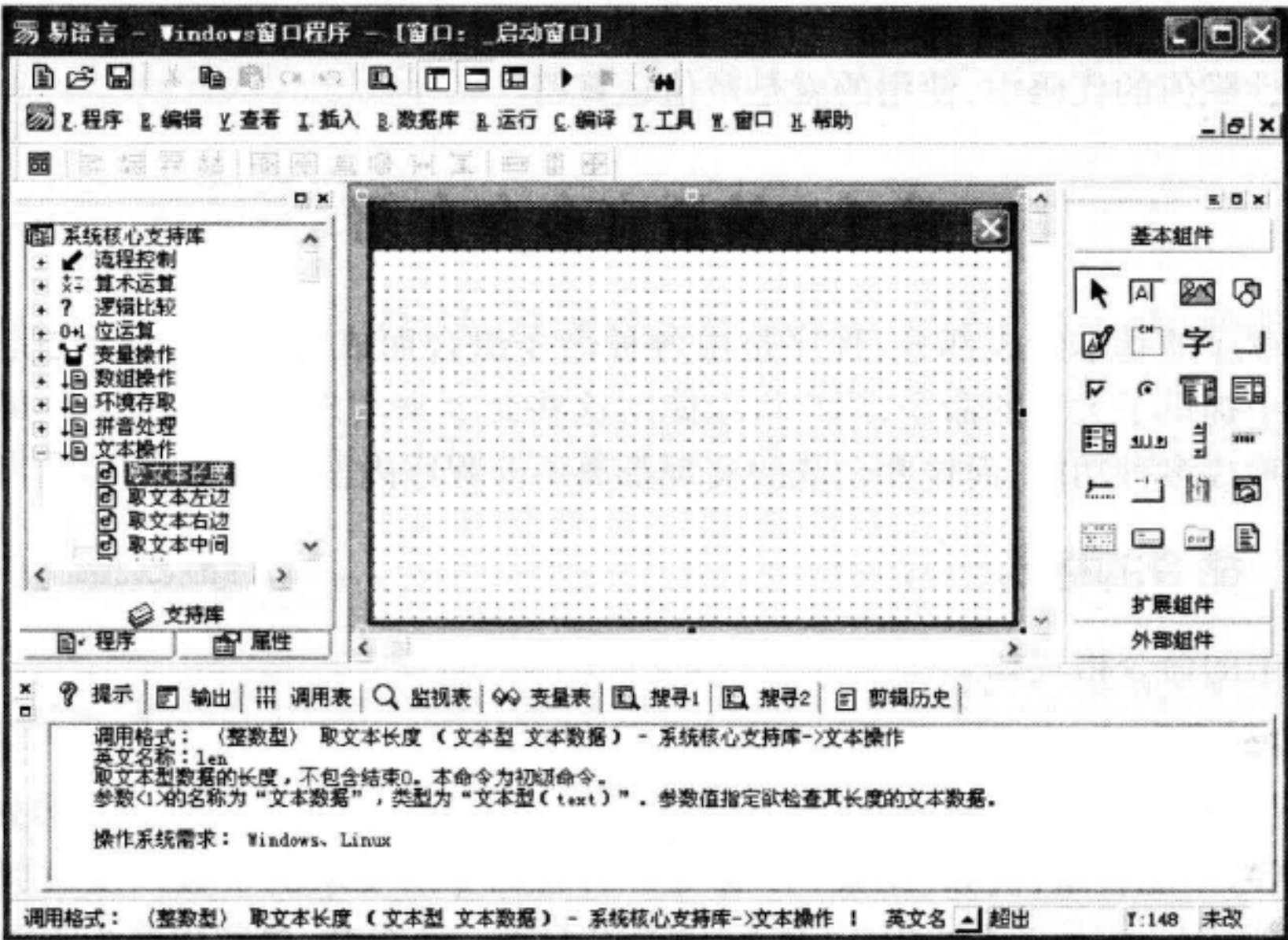


图 2-17 查看即时帮助信息

在编写程序时,如果忘记了某个命令参数的含义,可以将光标停在该命令上,然后按下 F1 键,在提示面板上则能看到该命令的即时帮助信息。

点击“帮助”菜单→“易语言知识库”,可以查看完整的帮助文档。其内容相对即时帮助而言更加丰富完整,并且每个命令都配有简单的例程。

2.4.3 命令的返回值

大多数命令执行完毕后都有返回值。有的命令返回运算结果,如“求正弦()”命令,返回求得的正弦值;有的命令返回执行的结果,如“取文本左边()”命令,返回取出来的文本内容;有的命令返回运行是否成功的状态,如“创建目录()”命令,创建成功返回“真”,创建失败则

返回“假”等等。当前命令的返回值对后续命令而言,往往是非常重要的。例如一个命令如果运行成功,就会弹出信息框提示成功,否则提示失败,如图 2-18 所示。

各命令的语法规定了其返回值的数据类型,在实际使用中,要注意有可能需要对返回值的数据类型加以转换,例如,编辑框的内容属性只能接收文本型数据,因此要显示一个数字时,就可以使用“到文本()”命令将数字转换为文本形式后赋值给编辑框显示,代码如下:

```
编辑框 1. 内容 = 到文本(求平方根(100))
```

有些命令的返回值是一个通用型的数据,根据参数不同,其返回值的数据类型也会不同。例如“多项选择(,)”命令,该命令有两个参数,第一个参数是索引值,第二个参数是待选项,待选项可以重复添加。待选项的数据类型是通用型的,表示参数 2 可以是任意的数据类型,那么返回哪个待选项取决于第一个参数的索引值。索引值为 1 表示返回第一个待选项;索引值为 2 则返回第二个待选项。所以,所选项是哪种类型的数据,返回值就为哪种类型的数据。

有些命令无返回值,例如“销毁()”命令,此类无返回值的命令运行后不返回任何值,所以直接使用即可。

命令是否有返回值,返回值的数据类型如何,都可以通过易语言的即时帮助系统查询到,在程序编辑界面,将光标停在欲查询的命令上,然后按下 F1 键,可以在提示面板中看到该命令的帮助。在提示面板中“调用格式”一行,写在命令名前面的就是该命令的返回值类型,如果无返回值则显示无返回值,如图 2-19 所示。

子程序名	返回值类型	公开	备注
按钮1_被单击			
判断 (编辑框1.内容 = “易语言”)			
信息框 (“输入正确”, 0 + #信息图标,)			
信息框 (“请输入正确的名称”, 0 + #警告图标,)			

图 2-18 命令返回值

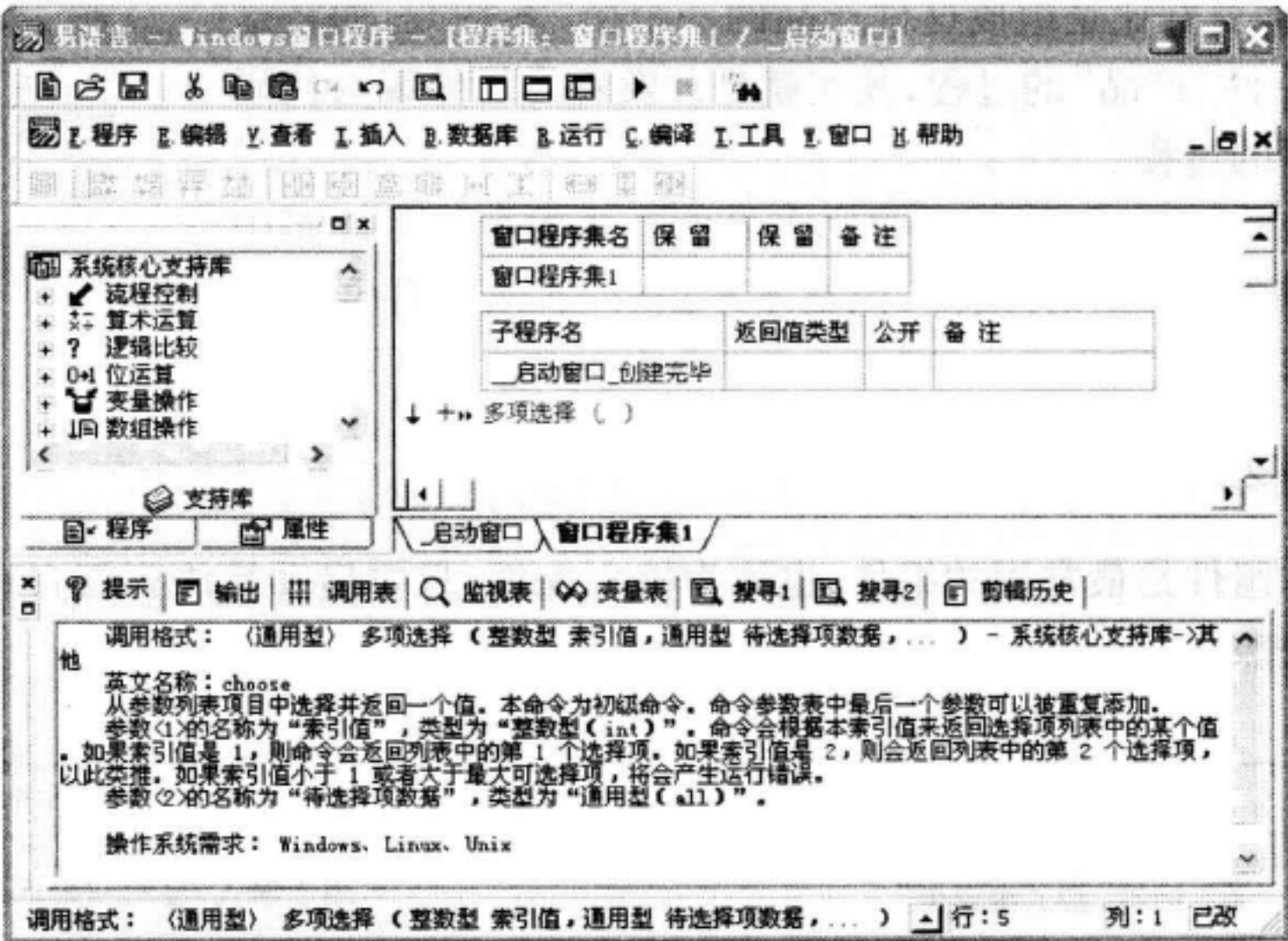


图 2-19 查看命令的返回值类型

2.4.4 命令的套用

易语言中的命令是可以套用的,即命令中包含命令。例如以下语句:

编辑框 1. 内容 = 到文本(到数值(编辑框 1. 内容) + 1)

此行代码在“到文本()”命令中套用了到数值命令,命令的嵌套调用是将一个命令的返回值作为另一个命令的参数。上述代码首先会将编辑框中的内容转换成数值型,然后将转换后的数值加 1,将其结果再转换成文本重新在编辑框中显示。

例如,“写到文件()”命令的参数中套用多个命令的代码如图 2-20 所示:

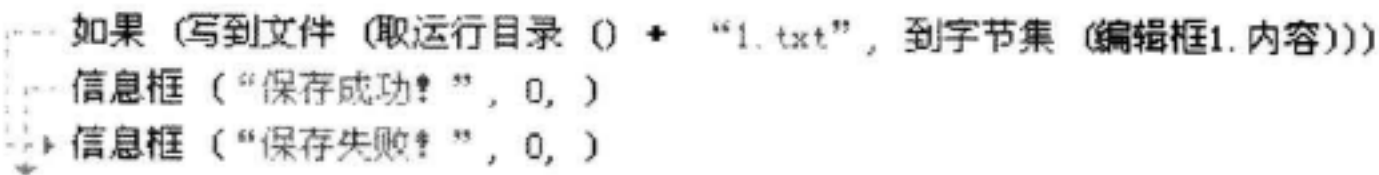


图 2-20 命令的套用

上述代码实现的功能是将编辑框中的内容保存为程序运行目录下的一个文本文件,并弹出信息框提示是否保存成功。

2.5 易语言组件介绍

易语言中各种常用组件已到达 80 个之多。这些组件涉及软件开发中各方面的应用。随着版本不断升级,组件数量还将不断地增加。

其中,窗口是一种特殊的组件,它可以作为其他组件的载体。这里所说的其他组件,和窗口一样,都是创建界面的基本结构模块。一个 Windows 窗口程序一般都要包含若干个不同的组件,相互组合分别实现不同的功能。如果将开发出的程序形象地比喻成一个产品的话,那么组件就好比是组成这件产品的一个个“零件”。而将这些“零件”组成一件“产品”的过程,其实就是开发者为组件编写代码和集成组件程序的过程。

在新建一个易程序后,可以在程序主界面右边的窗口组件箱中看到当前系统所支持的所有组件,包括基本组件、扩展组件和外部组件三类,如图 2-21 所示。



图 2-21 易语言组件箱

2.5.1 基本组件

易语言基本组件是最常用的组件,也称为核心控件,只要启动易语言就自动加载,不能删除。基本组件按照功能大致可分为 10 类,如表 2-6 所示。

表 2-6 基本组件分类表

基本组件分类名称	基本组件图标和名称	基本组件用途
文字显示组件	字“标签”“编辑框”	用于输入,显示和编辑单项文本数据
列表类组件	“组合框”“列表框”“选择列表框”	显示多条记录和数据
按钮组件	“按钮”“单选框”“多选框”“超级链接框”	主要用于鼠标点击时执行各种操作
文件系统组件	“驱动器框”“目录框”“文件框”	主要用于对计算机硬盘中的文件进行管理

(续)

基本组件分类名称	基本组件图标和名称	基本组件用途
图形处理和多媒体组件	“画板”“图片框”“颜色选择器”“外形框” “影像框”	用于对图形文件和多媒体文件的显示和处理
网络组件	“数据报”“服务器”“客户”“端口”	主要用于网络之间的通信
滚动组件	“横向滚动条”“纵向滚动条”“进度条”“滑块条”“调节器”	主要用于不能自动提供滚动条的组件
时间组件	“日期框”“月历”“时钟”	用于显示时间和日期
数据库组件	“表格”“数据源”“通用提供者”“数据库提供者”“外部数据库”“外部数据库提供者”	用于处理内部和外部数据库数据
其他类别组件	“分组框”“通用对话框”“打印机”“选择夹”	

2.5.2 扩展组件

易语言扩展组件大多是其他支持库提供的组件,常用标准扩展组件分类如表 2-7 所示。

表 2-7 扩展组件分类表

扩展组件分类名称	扩展组件图标和名称	扩展组件用途
文字显示组件	“透明标签”“超级编辑框”	用于输入、显示和编辑单项文本数据
列表类组件	“树形框”“超级列表框”	显示多条记录和数据
按钮组件	“工具条”“超级按钮”	主要用于鼠标单击执行各种操作
多媒体组件	“高级影像框”	用于多媒体文件的显示和处理
网络组件	“超文本浏览框”“IP 编辑框”	主要用于网络之间的通信
数据库组件	“数据库连接”“记录集”	用于处理内部和外部数据库数据
其他类别组件	“分隔条”“状态条”	

除了基本组件和扩展组件以外,还有外部组件。外部组件是由注册到易语言中的 OCX 或类型库自动增加的。常用的外部控件有 Windows 媒体播放器。

组件分类还可以根据其可否容纳其他组件,划分为容器类和非容器类;或按运行时是否具有可视外形划分为界面类和功能类。容器类组件内可以包含其他的组件,如选择夹、图片框、外形框等组件;而功能类组件仅用作提供某种功能,运行时不可见,如时钟、打印机等组件。

易语言为程序开发人员提供的组件已经相当全面,只要在编程中合理地使用这些组件,灵活搭配,就能使程序同时拥有漂亮的界面和强大的功能,并且能提高程序开发的效率。

大部分组件都拥有自己的属性、方法和事件,也就是说组件是属性、方法和事件的集合。组件也被称为“对象”,比如说,一个人、一件东西、一件事情,都可以被认为是一个“对象”。对象的属性记录着对象的特征,对象的方法提供了操作对象的途径,对象的事件用于通知外部其状态发生了改变。比如说一个电源开关,其外形、颜色、使用电压等都可以认为是该电源开关“对象”的属性,而关闭或打开此电源开关则可以认为是电源开关“对象”的方法。在关闭或打开的同时,它可能产生事件,如通过与其相连的电器开始工作或停止等。易语言程序中的“画板”组件就是一个典型的对象,它具有“画笔类型”、“画笔粗细”等属性,“画直线”、“画矩形”等方法,在被重新绘画时还会产生“绘画”事件。

2.6 成员属性、方法与事件

2.6.1 通用属性

1. 控件的位置及尺寸

所有可视化的控件都有以下 4 个属性,它们决定着控件的位置和大小。

- 左边:返回或设置控件左边与其容器的左边之间的距离,即控件相对于容器的 X 坐标。
- 顶边:返回或设置控件顶边与其容器的顶边之间的距离,即控件相对于容器的 Y 坐标。
- 宽度:返回或设置控件的宽度。
- 高度:返回或设置控件的高度。

在“左边”和“顶边”属性中都提到了容器。当控件包含在窗体或另一个控件之内,窗体或另一个控件就称为该控件的容器。容器内的控件坐标不是绝对坐标,而是缺省以容器左上角为坐标原点的相对坐标。

2. 控件的颜色和字体

- 文本颜色:返回或设置控件里文本的颜色。
- 背景颜色:返回或设置控件的背景颜色。

3. 控件的值

一般控件都有属性来存放最重要或最常用的数据,此属性称为控件的值。大多数控件的值会选用标题或者内容属性。

- 标题属性:窗口、标签、按钮等控件用来设置或者返回控件中的静态文本。
- 内容属性:编辑框等控件用来设置或返回控件中的动态文本。
- 可视:返回或设置一个控件是否可见,取值为逻辑型。
- 禁止:返回或设置一个控件是否可以使用,取值为逻辑型。

例如,当按钮.禁止 = 真时,该按钮控件将以灰色显示,表示该控件无效,不响应操作。而当某个控件的可视属性值为假时,在程序界面中将看不到该控件。

2.6.2 通用方法

1. 移动

该方法可以移动控件或改变控件的大小,英文名称为 move。使用移动方法比直接赋值的效率高。其调用格式如下:

<无返回值> 对象.移动([整数型 左边],[整数型 顶边],[整数型 宽度],[整数型 高度])

参数<1>的名称为“左边”,类型为“整数型(int)”,可以被省略,单位为像素点。若该参数被省略则不改变左边原位置。

参数<2>的名称为“顶边”,类型为“整数型(int)”,可以被省略,单位为像素点。若该参数被省略则不改变顶边原位置。

参数<3>的名称为“宽度”,类型为“整数型(int)”,可以被省略,单位为像素点。若该参数被省略或等于-1,则不改变该控件的原宽度。

参数<4>的名称为“高度”,类型为“整数型(int)”,可以被省略,单位为像素点。若该参数被省略或等于-1,则不改变该控件的原高度。

2. 获得焦点

所谓“输入焦点”,即当前用户按键操作所对应的目标对象。比如,用户正在编辑框中输入文本,那么该编辑框就具有输入焦点,因为它将获得目前用户所进行的所有按键操作。所谓“获得”或者“失去”输入焦点,即当前用户按键操作所对应的目标对象发生转移。比如,用户当前正在编辑框1总输入文本,突然使用鼠标或其他方法切换到编辑框2上去继续输入,那么此时,编辑框1就“失去”了输入焦点,而编辑框2则“得到”了输入焦点,因此在此以后用户所进行的所有按键操作都将被编辑框2所获得。

2.6.3 成员事件

事件就是发生的事情,有外界的刺激(键盘或者鼠标的输入),也有内部的变化,例如时钟的周期事件。或者说事件就是对象能识别的一个动作或内部状态的改变。事件过程是响应该事件要执行的代码。只有了解清楚控件能够识别的事件后,才能编写出正确有效的应用程序。

大多数控件都能识别的一些通用事件有以下几种:

1. 单击和双击事件

被单击:当在一个控件上按下鼠标按钮时,发送单击事件。

被双击:当鼠标快速按下并放开2次时会发生的事件。

2. 键盘事件

按下某键:当按下一个对应某ASCII字符键时,触发该事件。

放开某键:当放开一个对应某ASCII字符键时,触发该事件。

3. 焦点事件

获得焦点:当窗口或控件获得焦点时触发该事件。

失去焦点:当窗口或控件失去焦点时触发该事件。

易语言的窗口和控件都有一个预定的事件集,即可识别的事件。虽然对象能够识别的事件种类很多,但并不代表所有发生的事件都会引起对象反应。比如按钮控件一般只响应单击事件,同时它也能识别鼠标移动事件,但鼠标移动相对一个按钮控件来说意义并不大,可以对此事件不作响应。

总而言之,编写可视化应用程序的主要任务是,判断控件是否响应某事件及其如何响应该事件,当想让控件响应某事件时,就把代码写到这个事件过程中。

第3章 易语言基本命令

为了达到实用计算机进行计算和解决问题的目的,必须将问题的解决方案用计算机指令表达出来。也就是说,必须提供给计算机一套指令,才能使其方便地解决问题,这一套指令就是一段程序。将解决问题的一段程序进行封装即为命令。一个命令可以由一个或多个简单的命令来封装成一个功能更强大的命令。当需要相同作用的功能时,只要调用新的命令便可以得到相同的结果,而且还大大简化了编程的复杂度,缩短了软件的开发周期。

命令是软件的基本组成部分,一个软件是由各种命令组合而成的,不同的命令完成不同的工作。在易语言中提供了大量的命令,用户可以使用这些命令来实现预想的运行效果。

3.1 流程控制

程序的流程分为3种类型:顺序结构、条件分支结构和循环结构。

顺序结构是最简单的,只要语句一行接一行地执行就是顺序结构。条件分支结构就是当一个条件满足时产生一个动作,不满足时产生另外的动作。循环结构是当条件不满足要求时,重复上个动作,直到条件满足要求时则跳出循环。

流程控制类命令可以控制程序运行的路线,大多数程序的编写都离不开这类命令,是易语言中非常重要的一类命令。

流程控制类命令分为三类:分支类流程控制命令、循环类流程控制命令和跳转类流程控制命令。

3.1.1 分支类流程控制命令

分支类流程控制命令包括如果()、如果真()、判断()。

1. “如果()”命令

命令格式为: <无返回值> 如果(逻辑型 条件)

(1) “如果()”命令的参数“条件”为一个逻辑型的数据。根据提供的逻辑参数的值来决定是否改变程序的执行位置。如果条件为真,则程序顺序执行后续代码;若条件为假,则程序跳转到沿虚线箭头所示的代码行继续运行。

(2) 【范例3-1】新建一个易程序,在“_启动窗口”中添加一个编辑框组件和两个按钮组件,如果编辑框中有“你好易语言!”的字样,按下确定按钮,就会出现一段其他的话,否则,编辑框就会显示“你好易语言!”的字样。具体参考例程3-1。

(3) 启动窗口程序集如图3-1所示。

(4) 【运行结果】运行例程3-1,观测单击确定按钮前后编辑框显示内容的变化,如图3-2所示。

(5) 【范例解析】按F5键运行程序,此时编辑框中没有任何内容,单击确定按钮,可以看

子程序名	返回值类型	公开	备注
__启动窗口_创建完毕			

子程序名	返回值类型	公开	备注
__确定按钮_被单击			
如果 (编辑框1.内容 = "你好易语言!")			
编辑框1.内容 = "欢迎使用易语言,谢谢您的支持,我们会做得更好!"			
编辑框1.内容 = "你好易语言!"			

子程序名	返回值类型	公开	备注
__取消按钮_被单击			
启动窗口.销毁()			

图 3-1 “如果()”命令代码示例

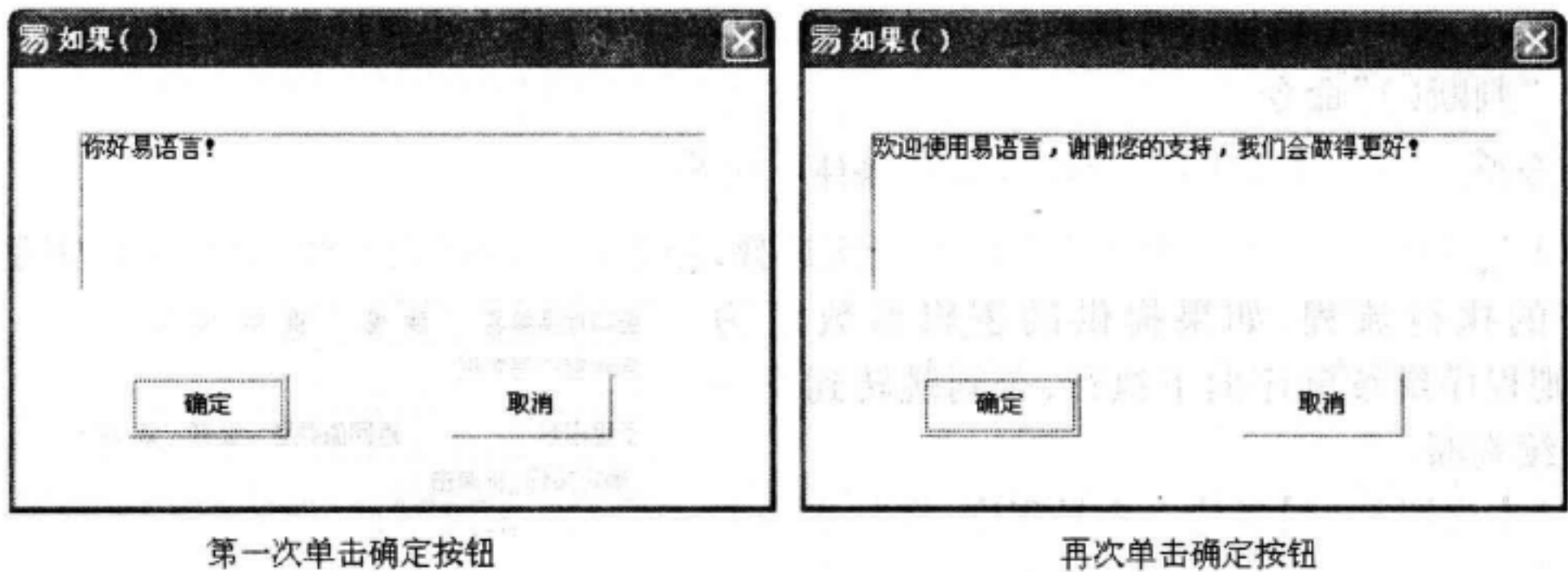


图 3-2 “如果()”命令运行结果示例

到编辑框显示“你好易语言!”;再单击确定按钮,此时编辑框中已有“你好易语言!”字样,因此编辑框显示“欢迎使用易语言,谢谢您的支持,我们会做得更好!”这段文本。

2. “如果真()”命令

命令格式为:<无返回值>如果真(逻辑型 条件)

(1) “如果真()”命令的参数“条件”也是一个逻辑型的数据,根据提供的逻辑参数的值,来决定是否改变程序的执行位置,如果提供的逻辑参数值为真,程序继续顺序向下执行,否则跳转到左侧箭头指向的命令处去执行。与“如果()”命令不同的是,“如果真()”命令没有为逻辑参数为假时的程序部分。

(2) 【范例 3-2】新建一个易程序,在“__启动窗口”中添加一个编辑框组件和一个按钮组件,如果编辑框中有任何内容,即不为空,按下确定按钮,就会出现相应的效果,否则不会出现任何变化。具体参考例程 3-2。

(3) 启动窗口程序集,如图 3-3 所示。

(4) 【运行结果】运行例程 3-2,观测编辑框的内容是否为空的情况下,单击确定按钮后编辑框多项属性的变化,如图 3-4 所示。

(5) 【范例解析】按 F5 键运行程序,此时编辑框中没有任何内容,单击确定按钮,可以看到界面没有任何变化;当在编辑框中输入任意文本后再单

窗口程序集名	保留	保留	备注
启动窗口程序集			

子程序名	返回值类型	公开	备注
__确定按钮_被单击			
如果真 (编辑框1.内容 != "")			
编辑框1.背景颜色 = #深青			
编辑框1.文本颜色 = #浅灰			
编辑框1.字体.字体大小 = 29			
编辑框1.内容 = "欢迎您使用易语言"			

图 3-3 “如果真()”命令代码示例

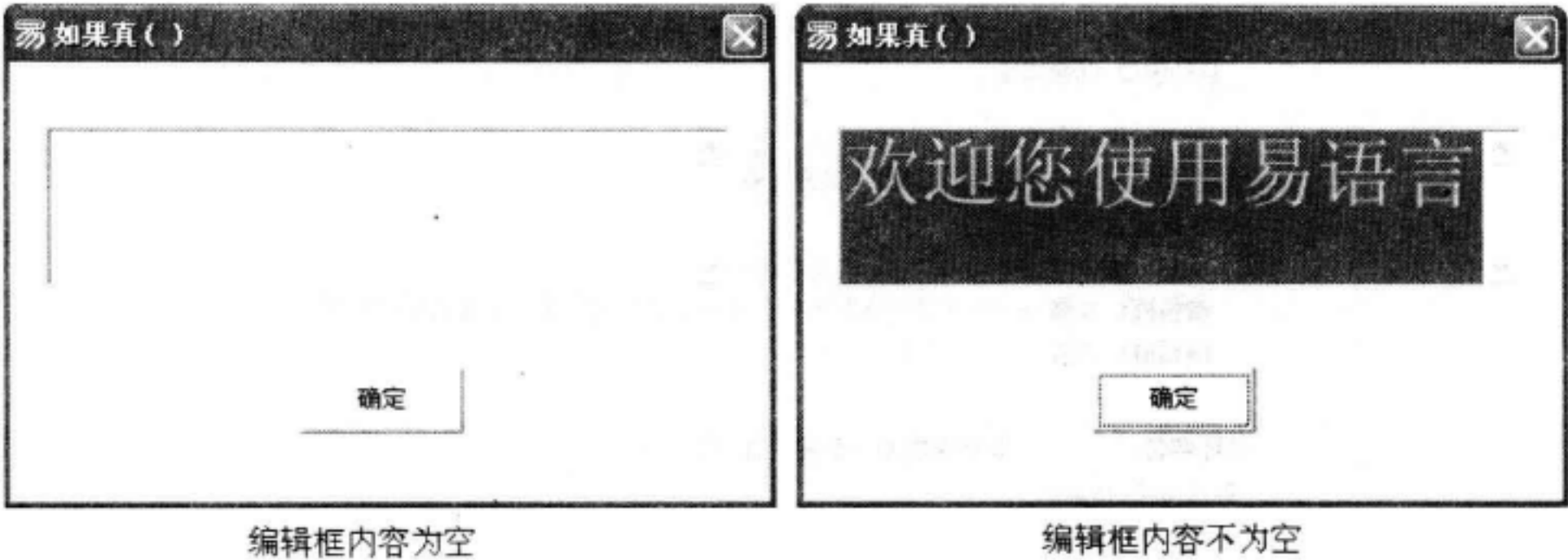


图 3-4 “如果真()”命令运行结果示例

击确定按钮,此时编辑框的背景颜色、文本颜色、内容以及字体大小属性发生了变化。

3. “判断()”命令

命令格式为: <无返回值> 判断(逻辑型 条件)

(1) “判断()”命令主要用于条件的分支选择,根据提供的逻辑参数的值,来决定是否改变程序的执行流程,如果提供的逻辑参数值为“真”,则程序继续顺序向下执行,否则跳转到下一分支继续判断。

(2) 【范例 3-3】新建一个易程序,在“_启动窗口”中添加一个标签组件、一个编辑框组件和一个按钮组件,如果编辑框中的内容为“易语言”,按下确定按钮,就会弹出“欢迎使用易语言”的显示框,否则会出现对应错误提示的显示框。具体参考例程 3-3。

(3) 启动窗口程序集,如图 3-5 所示。

(4) 【运行结果】运行例程 3-3,观测编辑框内输入不同内容时弹出提示信息框的变化,如图 3-6 所示。

窗口程序集名	保留	保留	备注
启动窗口程序集			

子程序名	返回值类型	公开	备注
确定按钮_被单击			

```
判断 (编辑框1.内容 = “易语言”)
信息框 (“欢迎使用易语言”, 0, )
※提示信息: “欢迎使用易语言”
※按钮: 0
※窗口标题:
判断 (编辑框1.内容 = “”)
信息框 (“用户名不能为空”, 48, )
判断 (编辑框1.内容 ≠ “易语言”)
信息框 (“请输入正确的用户名”, 16, )
```

图 3-5 “判断()”命令代码示例

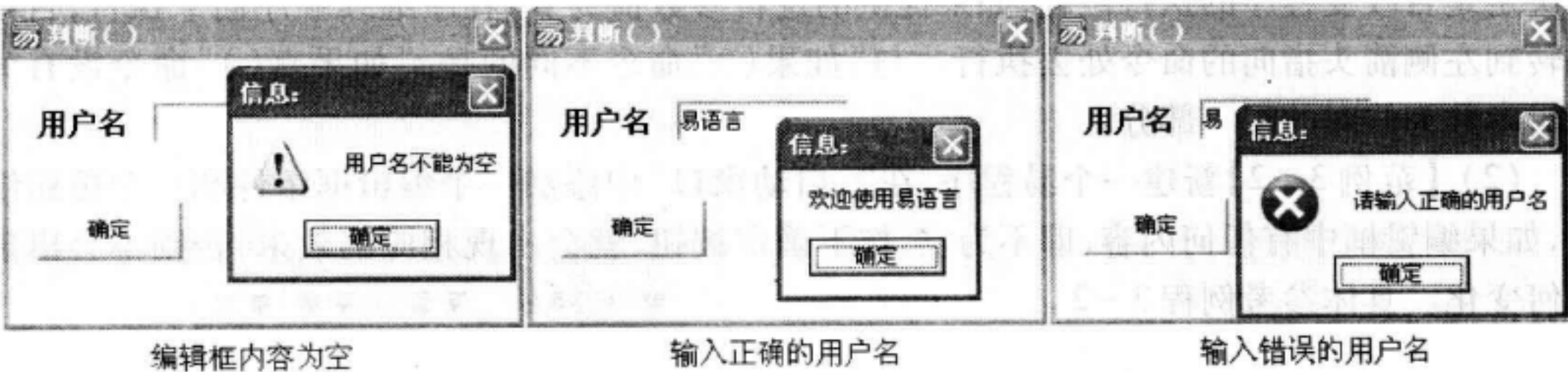


图 3-6 “判断()”命令运行结果示例

(5) 【范例解析】按 F5 键运行程序,此时编辑框中没有任何内容,单击确定按钮,弹出“用户名不能为空”的警告信息框;当在编辑框中输入“易语言”再单击确定按钮,弹出“欢迎使用易语言”的提示信息框;当在编辑框中输入非“易语言”的任何内容时,弹出的则是“请输入正确用户名”的错误信息框。

3.1.2 循环类流程控制命令

循环流程类命令都由循环首和循环尾组成,二者是成对出现,不会单一存在。当在代码中输入了循环首命令,循环尾就会自动出现。循环首表示循环的开始,循环尾表示循环的结束,二者之间的代码称为循环块,是循环类命令重复执行的代码。循环类流程控制命令就是在一定条件下多次执行循环块,从而减轻了程序员的工作量。

循环类流程类控制命令包括判断循环首()、循环判断首()、计次循环首()、变量循环首()。

1. “判断循环首()”

命令格式为: <无返回值> 判断循环首()

(1) “判断循环首()”命令根据提供的逻辑参数的值,来决定是否进入循环。如果提供的逻辑参数值为真,程序顺序执行下一条命令进入循环。每次循环结束后,会再一次检查“判断循环首”中的条件成立,如果条件不成立了,或者提供的逻辑参数的初始值为假,则退出循环,直接跳转到本命令所对应的“判断循环尾”的下一条命令处。

(2) 【范例3-4】新建一个易程序,在“_启动窗口”中添加一个标签组件、一个编辑框组件和一个按钮组件,单击按钮组件,在编辑框中显示出100以内的偶数,运用“判断循环首”命令实现。具体参考例程3-4。

(3) 启动窗口程序集,如图3-7所示。

(4) 【运行结果】运行例程3-4,观测单击按钮后编辑框内显示的内容,如图3-8所示。

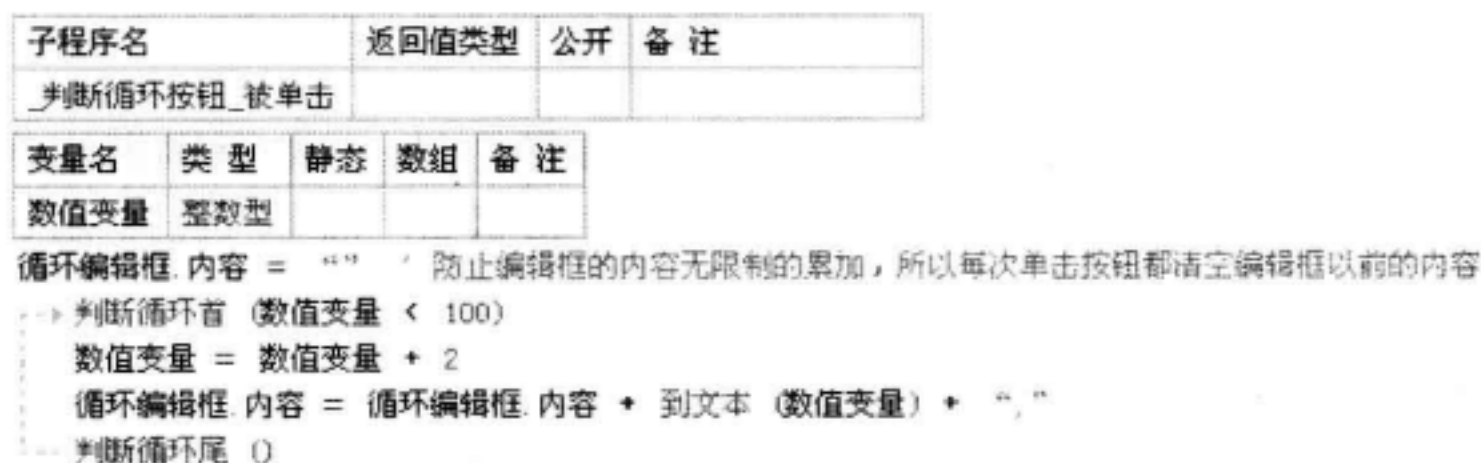


图 3-7 “判断循环首()”

命令代码示例



图 3-8 “判断循环首()”

命令运行结果示例

(5) 【范例解析】按 F5 键运行程序,此时编辑框中没有任何内容,单击显示按钮,编辑框中会依次显示出100以内的所有偶数。

2. “循环判断首()”命令

命令格式为: <无返回值> 循环判断首()

(1) “循环判断首()”命令首先执行一次循环块,然后再判断循环条件是否成立,如果“循环判断首()”中的条件为“真”,跳到循环首处继续新一次的循环;反之,如果条件不成立,则循环终止,向下执行其他代码。

“循环判断首()”命令和“判断循环首()”命令相似,不同点在于,“判断循环首()”命令是在循环首中判断,先判断再循环,符合条件就执行循环块,不符合条件就停止循环;而“循环判断首()”命令是先循环再判断,无论条件是否满足,先执行一次循环块,然后在循环尾部判断,如果条件成立,跳到循环首,顺序执行循环块中的命令。

(2) 【范例3-5】新建一个易程序,在“_启动窗口”中添加一个标签组件、一个编辑框组件

和一个按钮组件,单击按钮组件,在编辑框中显示出 100 以内的偶数,运用“循环判断首”命令实现。具体参考例程 3-5。

(3) 启动窗口程序集,如图 3-9 所示。

(4) 【运行结果】运行例程 3-5,观测单击按钮后编辑框内显示的内容,如图 3-10 所示。

子程序名	返回值类型	公开	备注
循环判断按钮_被单击			

变量名	类型	静态	数组	备注
数值变量	整数型			

循环编辑框.内容 = "" 防止编辑框的内容无限制的累加,所以每次单击按钮都清空编辑框以前的内容

```

循环判断首 ()
    数值变量 = 数值变量 + 2
    循环编辑框.内容 = 循环编辑框.内容 + 到文本 (数值变量) + ","
循环判断尾 (数值变量 < 100)
  
```

图 3-9 “循环判断首()”

命令代码示例



图 3-10 “循环判断首()”

命令运行结果示例

(5) 【范例解析】按 F5 键运行程序,此时编辑框中没有任何内容,单击显示按钮,编辑框中会依次显示出 100 以内的所有偶数。

3. “计次循环首()”命令

命令格式为: <无返回值>计次循环首(整数型 循环次数,[整数型变量 已循环次数记录变量])

(1) “计次循环首()”命令的第一个参数指定循环的次数,是整数型的,第二个参数是可选的,可以填入一个整数型的变量,用来记录已经循环的次数,第一次循环时变量值为 1,第二次循环时变量值为 2,依此类推。

(2) 【范例 3-6】新建一个易程序,在“_启动窗口”中添加一个标签组件、一个编辑框组件和一个按钮组件,单击按钮组件,在编辑框中显示出 1~100 的所有整数和,运用“计次循环首”命令实现。具体参考例程 3-6。

(3) 启动窗口程序集,如图 3-11 所示。

(4) 【运行结果】运行例程 3-6,观测单击计算按钮后编辑框内显示的内容,如图 3-12 所示。

子程序名	返回值类型	公开	备注
显示结果按钮_被单击			

变量名	类型	静态	数组	备注
累加变量	整数型			
计次变量	整数型			

```

计次循环首 (100, 计次变量)
    累加变量 = 累加变量 + 计次变量
计次循环尾 ()
编辑框1.内容 = 到文本 (累加变量)
  
```

图 3-11 “计次循环首()”命令代码示例

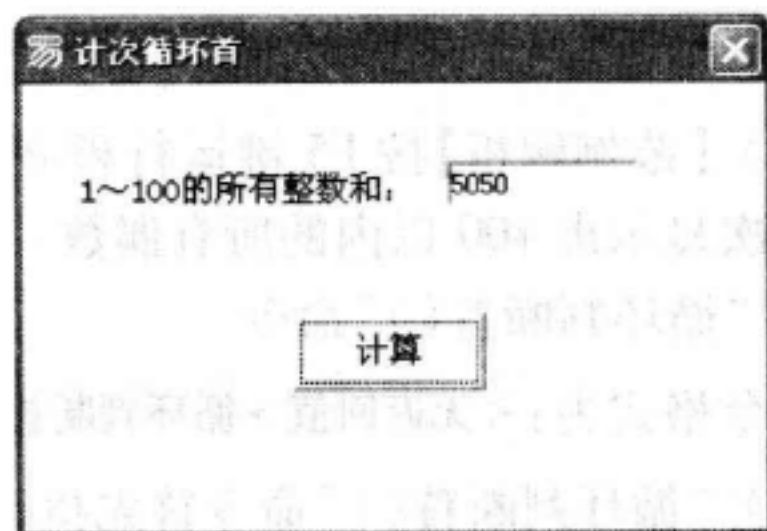


图 3-12 “计次循环首()”命令运行结果示例

(5) 【范例解析】按 F5 键运行程序,此时编辑框中没有任何内容,单击计算按钮,编辑框中会显示出 1~100 的所有整数和为 5050。在计次循环首内部每次循环让“累加变量”的数值等于它当前的数值加上循环的次数,此范例中的循环次数从 1 开始到 100 结束。

4. “变量循环首()”命令

命令格式为: <无返回值>变量循环首(整数型 变量起始值,整数型 变量目标值,整数型 变量递

增值,[整数型变量 循环变量])

(1) “变量循环首()”命令用于一个变量的内部循环,有四个参数,定义了变量的起始值、目标值和递增值,每经过一次循环,变量的起始值都会按指定的递增值增加,一旦该变量的值超过目标值,则退出循环。第四个参数可以传递一个变量,来接收循环时变量的当前值。

(2) 【范例3-7】新建一个易程序,在“_启动窗口”中添加两个标签组件、两个编辑框组件和一个按钮组件,单击按钮组件,在两个编辑框中分别显示出200~400之间所有整数和以及循环结束时变量的终值,运用“变量循环首”命令实现。具体参考例程3-7。

(3) 启动窗口程序集,如图3-13所示。

(4) 【运行结果】运行例程3-7,观测单击计算按钮后编辑框内显示的内容,如图3-14所示。

子程序名	返回值类型	公开	备注
显示结果按钮_被单击			

变量名	类型	静态	数组	备注
变量	整数型			
累加变量	整数型			

```

变量循环首 (200, 400, 1, 变量)
累加变量 = 累加变量 + 变量
变量循环尾 0
编辑框1.内容 = 到文本 (累加变量)
编辑框2.内容 = 到文本 (变量)

```

图3-13 “变量循环首()”命令代码示例

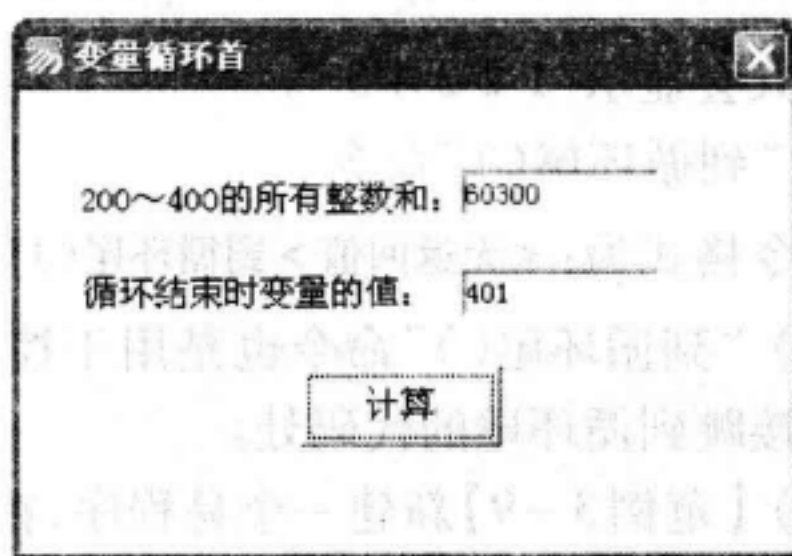


图3-14 “变量循环首()”命令运行结果示例

(5) 【范例解析】按F5键运行程序,此时编辑框中没有任何内容,单击计算按钮,编辑框中会显示出200~400之间所有整数的和为60300。此范例中,随着循环的进行,变量的值逐次增加1,直到增加到400时,与累加变量进行加法运算后自增到401,变量的值超过了目标值400,循环到此结束,此时的“累加变量”就是所求的和,其结果显示在编辑框中。

3.1.3 跳转类流程控制命令

跳转类流程控制命令可以使得程序的流程控制更加方便。“跳出循环()”和“到循环尾()”命令可以控制循环的执行流程,“返回()”和“结束()”命令可以控制子程序和整个程序的执行流程。

1. “跳出循环()”命令

命令格式为: <无返回值>跳出循环()

(1) “跳出循环()”命令用于控制循环。当一个循环中运行了“跳出循环()”命令,那么当前的循环就会结束,然后继续运行循环体以后的代码。如果在循环过程中,当某个条件满足,想提前结束该循环,就可以使用“跳出循环()”命令。

(2) 【范例3-8】新建一个易程序,在“_启动窗口”中添加一个标签组件、一个编辑框组件和一个按钮组件,单击按钮组件,在编辑框中只显示不超过5的整数,运用“跳出循环”命令实现。具体参考例程3-8。

(3) 启动窗口程序集,如图3-15所示。

(4) 【运行结果】运行例程3-8,观测单击显示按钮后编辑框内显示的内容,如图3-16所示。

子程序名		返回值类型	公开	备注
_显示结果按钮_被单击				
变量名	类型	静态	数组	备注
循环变量	整数型			
计次循环首 (10, 循环变量)				
如果真 (循环变量 > 5)				
跳出循环 0				
编辑框1.加入文本 (到文本 (循环变量) + " ")				
计次循环尾 0				

图 3-15 “跳出循环()”命令代码示例

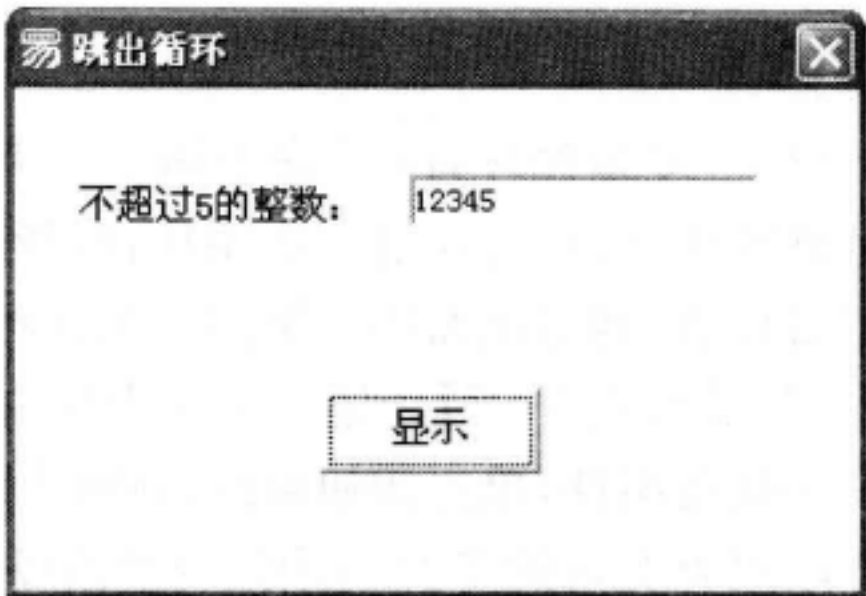


图 3-16 “跳出循环()”命令运行结果示例

(5) 【范例解析】当循环变量等于 6 时,满足了“如果真”的判断,将执行“跳出循环()”命令,即使循环次数还未达到“计次循环首()”命令中设定的 10 次,循环也会立即结束,所以编辑框中只会显示“1 2 3 4 5”。

2. “到循环尾()”命令

命令格式为:<无返回值>到循环尾()

(1) “到循环尾()”命令也是用于控制循环。当一个循环中运行了“到循环尾()”命令,将会直接跳到循环尾的代码处。

(2) 【范例 3-9】新建一个易程序,在“_启动窗口”中添加一个标签组件、一个编辑框组件和一个按钮组件,单击按钮组件,在编辑框中只显示 10 以内的偶数,运用“到循环尾”命令实现。具体参考例程 3-9。

(3) 启动窗口程序集,如图 3-17 所示。

(4) 【运行结果】运行例程 3-9,观测单击显示按钮后编辑框内显示的内容,如图 3-18 所示。

子程序名		返回值类型	公开	备注
_显示结果按钮_被单击				
变量名	类型	静态	数组	备注
循环变量	整数型			
计次循环首 (10, 循环变量)				
如果真 (循环变量 % 2 = 1)				
到循环尾 0				
编辑框1.加入文本 (到文本 (循环变量) + " ")				
计次循环尾 0				

图 3-17 “到循环尾()”命令代码示例

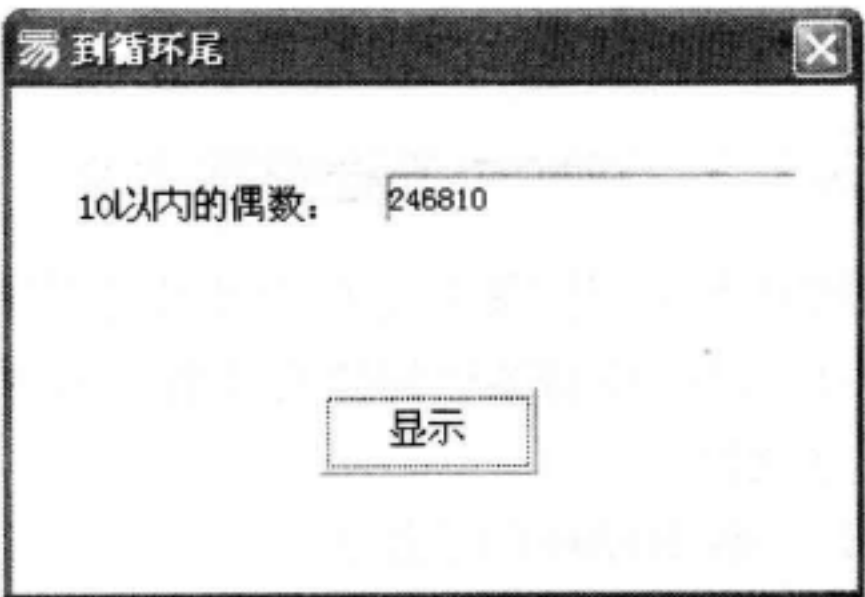


图 3-18 “到循环尾()”命令运行结果示例

(5) 【范例解析】当循环变量不能被 2 整除,即循环变量为奇数时,就会执行“到循环尾()”命令,跳过显示到编辑框的这句代码,只有循环变量为偶数时,才会执行将该变量显示到编辑框,所以编辑框中只会显示“2 4 6 8 10”。

3. “返回()”命令

命令格式为:<无返回值>返回([通用型 返回到调用方的值])

(1) “返回()”命令被执行后,会退出当前子程序。当前子程序中“返回()”命令之后的代码将不再被执行,程序将自动跳转到调用本子程序语句的下一条语句处继续执行。

由于“返回()”命令后面的代码不被运行,所以“返回()”命令也经常用于程序的调试。

有返回值的子程序必须使用“返回()”命令来返回执行的结果,需要注意的是实际返回值的数据类型要和子程序中定义的返回值数据类型相匹配。

(2)【范例3-10】类同范例3-8,在编辑框中只显示不超过5的整数,运用“返回()”命令来实现。具体参考例程3-10。

(3) 启动窗口程序集,如图3-19所示。

(4)【运行结果】运行例程3-10,观测单击显示按钮后编辑框内显示的内容,如图3-20所示。

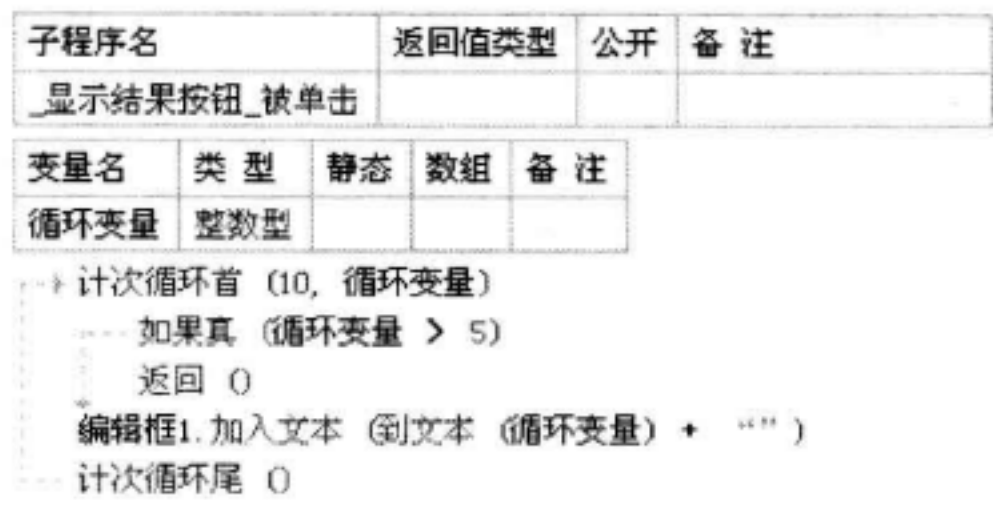


图 3-19 “返回()”命令代码示例

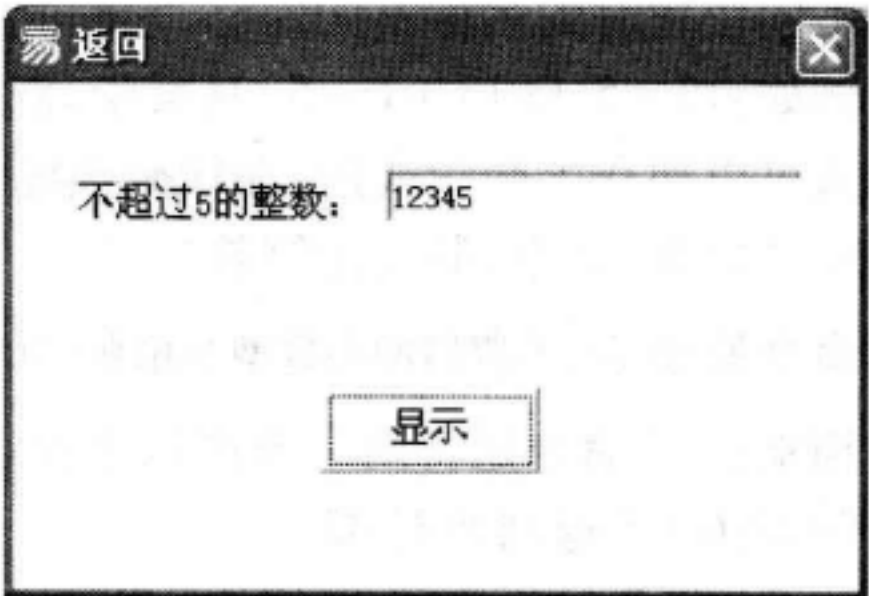


图 3-20 “返回()”命令运行结果示例

(5)【范例解析】当循环变量等于6时,满足了“如果真”的判断条件,将执行“返回()”命令来终止该循环的运行,使得当前循环中和当前子程序中的后续代码都不会被执行,所以编辑框中也只会显示“1 2 3 4 5”。

4、“结束()”命令

命令格式为: <无返回值> 结束()

“结束()”命令用于结束当前易程序的运行,可以实现程序的关闭功能。“结束()”命令没有返回值也没有参数。若仅仅只想关闭当前的某一个窗口,应该使用“窗口”的“销毁()”命令。建议尽可能在程序中使用销毁“_启动窗口”的方法来结束程序,有利于数据的安全。

3.2 算术运算

3.2.1 基本算术运算命令

基本算术运算命令在程序中最常使用的命令之一,包括相加(+)、相减或负(-)、相乘(*)、相除(/)、整除(\)、求余数(%)。

1. “相加”命令:简化运算符“ + ”

命令原型为: <通用型> 相加(通用型 被加数或文本或字节集,通用型加数或文本或字节集, ...)

相加(+)命令是基本运算符中唯一一个适用于任何基础类型的运算符,“相加”命令在数字运算时用于数字相加;在非数字运算时起到连接数据的作用。

当在两个数字之间使用“相加”命令则是将它们以数学方式相加,并返回之和。

例如: 8 + 6 返回的是 14; 而“ABC” + “OPQ”, 这个表达式是将 ABC 与 OPQ 这两个文本连接, 而不是将它们以数学方式相加。

使用“相加”命令时需要注意, 数字与数字相加时, 在易语言内部是以双精度小数型进行

计算的。

2. “相减”命令与“负”命令:简化运算符“-”

命令原型为: <双精度小数型>相减(双精度小数型 被减数,双精度小数型 减数,...)

<双精度小数型>负(双精度小数型 数值)

“相减”(-)命令是将两个数值进行求差运算,属于双目运算符,而“负”命令则是改变一个数的符号,属于单目运算符。“相减”(-)命令与“负”命令的运算符同为“-”,二者只是使用在不同位置时的名称不同而已,它们的概念可以相互转换。

例如:13-5 与 13+(-5)从数学理论上可以等同对待,在易语言中同样如此。在易语言中输入可以简化的数学式后,在代码预编译时会自动将其简化。

3. “相乘”命令:简化运算符“*”

命令原型为: <双精度小数型>相乘(双精度小数型 被乘数,双精度小数型 乘数,...)

相乘(*)命令是将两个数进行求积运算。

例如:6*7 返回的是42。

4. “相除”命令:简化运算符“/”

命令原型为: <双精度小数型>相乘(双精度小数型 被除数,双精度小数型 除数,...)

相除(/)命令是将两个数进行求商运算。

例如:35/7 返回的是5。

5. “整除”命令:简化运算符“\”

命令原型为: <双精度小数型>整除(双精度小数型 被除数,双精度小数型 除数,...)

整除(\)命令会将两个数进行求商运算并舍弃小数部分。

例如:22\4 返回的是5。

6. “求余数”命令:简化运算符“%”

命令原型为: <双精度小数型>求余数(双精度小数型 被除数,双精度小数型 除数,...)

求余数(%)命令是计算出被除数与除数求商之后的余数。

例如:22%4 返回的是2。

【注意】“相除”、“整除”和“求余数”这3个命令使用时,除数不能为0,因为数学中除数为0是没有意义的,因此在易语言程序中如果除数为0将会引发程序错误。

3.2.2 扩展算术运算命令

1. “取符号”命令

命令原型为: <整数型>取符号(双精度小数型 欲取其符号的数值)

取符号命令用于获取数值或者数值型变量的正负符号,如果为正数则返回大于零;如果为负数,则返回小于零;若为0时则返回0。

例如:输出调试文本(取符号(-827.43))

运行后输出结果为:-1。

2. “取绝对值”命令

命令原型为: <双精度小数型>取绝对值(双精度小数型 欲取其绝对值的数值)

取绝对值命令用于获取数值或数值型变量的绝对值。

例如:输出调试文本(取绝对值(-827.43))

运行后输出结果为:827.43。

3. “取整”命令

命令原型为:<整数型>取整(双精度小数型 欲取整的小数)

取整命令用于获取数值或数值型变量的整数部分。但“取整”命令在对正小数和负小数的处理时有不同的效果。

当对一个正小数取整时,小数部分将会被完全舍去。

例如:对8.2取整。

输出调试文本(取整(8.2))

运行后输出结果为:8。

当对一个负小数取整时,如果有小数部分,则结果为原整数部分减1。

例如:对-8.2取整。

输出调试文本(取整(-8.2))

运行后输出结果为:-9。

4. “绝对取整”命令

命令原型为:<整数型>绝对取整(双精度小数型 欲取整的小数)

绝对取整命令用于获取一个数值或数值型变量的整数部分。“绝对取整”命令与“取整”命令不同之处在于,不论被取整的数是正数还是负数,其小数部分都将被完全舍去。

例如:对8.2绝对取整。

输出调试文本(绝对取整(8.2))

运行后输出结果为:8。

例如:对-8.2绝对取整。

输出调试文本(绝对取整(-8.2))

运行后输出结果为:-8。

5. “四舍五入”命令

命令原型为:<双精度小数型>四舍五入(双精度小数型 欲被四舍五入的数值,[整数型 被舍入的位置])

四舍五入命令用于获取一个数值或数值型变量的四舍五入值,被舍弃的位置通过它的参数指定。

参数<2>的位置起到指定舍弃位置的作用。当它是0时,意味着不保留小数,对十分位进行四舍五入;当它是大于0时,表示保留的小数位数,它的下一位小数将进行四舍五入;当它是小于0时,表示对整数部分的从某一位开始四舍五入,舍弃此参数的绝对值减1的整数位数(用0占位),小数部分将被抛弃。

例如:对384.529四舍五入,第二个参数设为-2(从小数点开始,向左2位)。

输出调试文本(四舍五入(384.529,-2))

得到的结果为:400。

6. “求次方”命令

命令原型为: <双精度小数型> 求次方(双精度小数型 欲求次方数值, 双精度小数型 次方数)

求次方命令用于获取一个数值或数值型变量的 N 次方。也可以使用分指数的概念使用该命令来开一个数的 N 次方。

例如: 求次方(16, 0.25) 即为开 16 的 4 次方。

输出调试文本(求次方(16, 0.25))

得到的结果为: 2。

7. “求平方根”命令

命令原型为: <双精度小数型> 求平方根(双精度小数型 欲求其平方根的数值)

求平方根命令用于获取一个数值或数值型变量的平方根。

例如: 求 49 的平方根。

输出调试文本(求平方根(49))

得到的结果为: 7。

8. “求正弦”命令

命令原型为: <双精度小数型> 求正弦(双精度小数型 欲进行计算的角)

求正弦命令用于获取一个弧度的正弦值。因为使用的单位是弧度值, 所以如果需要计算一个角度的正弦值时应该按照数学方法转换, 即将角度值乘以 $\pi/180$, 在易语言中为一个常量, 常量名为 #pi。

例如: 使用求正弦命令取 45° 角的正弦值。

输出调试文本(求正弦($45 \times \text{\#pi} \div 180$))

得到的结果为: $\sqrt{2}/2$ 。

【注意】在易语言中 #pi 的值被固定为 3.1415926535, 当进行精密计算时要注意误差精度。

9. “求余弦”命令

命令原型为: <双精度小数型> 求余弦(双精度小数型 欲进行计算的角)

求余弦命令用于获取一个弧度的余弦值。

例如: 使用求余弦命令取 45° 角的余弦值。

输出调试文本(求余弦($45 \times \text{\#pi} \div 180$))

得到的结果为: $\sqrt{2}/2$ 。

10. “求正切”命令

命令原型为: <双精度小数型> 求正切(双精度小数型 欲进行计算的角)

求正切命令用于获取一个弧度的正切值。

例如: 使用求正切命令取 45° 角的正切值。

输出调试文本(求正切($45 \times \text{\#pi} \div 180$))

得到的结果为: 1。

11. “求反正切”命令

命令原型为: <双精度小数型> 求反正切(双精度小数型 欲求其反正切值的数值)

求反正切命令用于通过一个正切值求出它所对应的弧度值。若要将弧度转换为角度,应该按照数学中的方法将弧度值乘以 180 再除以。在易语言中 π 一般使用 #pi 常量代替。

例如:使用求反正切命令取 45° 角的反正切值。

输出调试文本(求反正切($45 \times \text{\#pi} \div 180$))

得到的结果为:0.6657737500145。

12. “求自然对数”命令

命令原型为:<双精度小数型>求自然对数(双精度小数型 欲求其自然对数的数值)

求自然对数命令用于取以 e 为底的对数。e 在易语言中使用 #e 常量代替。#e 在易语言中的固定值为 2.718282,当在进行精密计算时要注意误差精度。

13. “求反对数”命令

命令原型为:<双精度小数型>求反对数(双精度小数型 欲求其反对数的数值)

求反对数命令用于取以 e 为底(自然对数的底)的某次方。在使用时要注意数据类型的范围,避免溢出。

14. “是否运算正确”命令

命令原型为:<逻辑型>是否运算正确(双精度小数型 欲校验的计算结果)

是否运算正确命令用于检查算术运算命令的返回结果是否是正常的。如果该数值正确有效,返回真;如果该数值是运算错误或者运算溢出后的结果,则返回假。

检查结果的有效范围为:“乘”、“除”、“求次方”、“求平方根”、“求正弦值”、“求余弦值”、“求正切值”、“求反正切值”、“求自然对数”、“求反对数”。

例如:求平方根 -1 是否运算正确。

输出调试文本(是否运算正确(求平方根(-1)))

得到的结果为:在输出面板中输出假,表示运算结果不正确。

15. “置随机数种子”命令

命令原型为:<无返回值>置随机数种子([整数型 欲置入的种子数值])

置随机数种子命令用于设置产生随机数的基数。如果调用时不给参数,默认是使用当前系统启动时间到当前时间的毫秒数作为基数。在取随机数之前调用此命令可在一定程度上避免取得的随机数相同或有规律。

16. “取随机数”命令

命令原型为:<整数型>取随机数([整数型 欲取随机数的最小值],[整数型 欲取随机数的最大值])

取随机数命令用于在指定范围中获取一个伪随机数。

例如:取 1 ~ 100 的随机数。

输出调试文本(取随机数(1,100))

得到的结果为:在输出面板中输出一个随机数,取值范围在 1 ~ 100 之间。

3.3 逻辑比较与位运算

3.3.1 逻辑比较

1. “等于”命令:运算符号为“=”或“==”

命令原型为: <逻辑型> 等于(通用型 被比较值, 通用型 比较值)

“等于”命令是用于比较两个数据是否一致。如果被比较值与比较值一致, 返回“真”; 否则, 返回“假”。

注意: 比较值的数据类型必须与被比较值一致或者可以相互转换。但双精度小数型与小数型无须转换即可直接进行比较。

2. “不等于”命令: 运算符号为“<>”或“!=”或“≠”

命令原型为: <逻辑型> 不等于(通用型 被比较值, 通用型 比较值)

“不等于”命令是用于比较两个数据是否不一致。如果被比较值与比较值不一致, 返回“真”; 否则返回“假”。

【注意】比较值的数据类型必须与被比较值一致或者可以相互转换。但双精度小数型与小数型无须转换即可直接进行比较。

3. “小于”命令: 运算符号为“<”

命令原型为: <逻辑型> 小于(通用型 被比较值, 通用型 比较值)

“小于”命令是用于比较一个数据是否小于另一个数据。如果是则返回“真”; 否则返回“假”。当使用“小于”命令用来比较两个文本时, 按照一一对应比较文本中每个字符的 ASCII 码的值; 当用来比较两个日期时间型数据时, 比较的顺序为年、月、日、时、分、秒, 各项以数值大小做比较。

【注意】比较值的数据类型必须与被比较值一致或者可以相互转换。但双精度小数型与小数型无须转换就可以直接进行比较。

4. “大于”命令: 运算符号为“>”

命令原型为: <逻辑型> 大于(通用型 被比较值, 通用型 比较值)

“大于”命令是用于比较一个数据是否大于另一个数据。如果是则返回“真”; 否则返回“假”。当使用“大于”命令用来比较两个文本时, 按照一一对应比较文本中每个字符的 ASCII 码的值; 当用来比较两个日期时间型数据时, 比较的顺序为年、月、日、时、分、秒, 各项以数值大小做比较。

【注意】比较值的数据类型必须与被比较值一致或者可以相互转换。但双精度小数型与小数型无须转换就可以直接进行比较。

5. “小于或等于”命令: 运算符号为“<=”或“≤”

命令原型为: <逻辑型> 小于或等于(通用型 被比较值, 通用型 比较值)

“小于或等于”命令是用于比较一个数据是否小于或者等于另一个数据。如果是则返回“真”; 否则返回“假”。当使用“小于或等于”命令用来比较两个文本时, 按照一一对应比较文本中每个字符的 ASCII 码的值; 当用来比较两个日期时间型数据时, 比较的顺序为年、月、日、时、分、秒, 各项以数值大小做比较。

【注意】比较值的数据类型必须与被比较值一致或者可以相互转换。但双精度小数型与小数型无须转换就可以直接进行比较。

6. “大于或等于”命令: 运算符号为“>=”或“≥”

命令原型为: <逻辑型> 大于或等于(通用型 被比较值, 通用型 比较值)

“大于或等于”命令是用于比较一个数据是否大于或者等于另一个数据。如果是则返回

“真”；否则返回“假”。当使用“大于或等于”命令用来比较两个文本时，按照一一对应比较文本中每个字符的 ASCII 码的值；当用来比较两个日期时间型数据时，比较的顺序为年、月、日、时、分、秒，各项以数值大小做比较。

【注意】比较值的数据类型必须与被比较值一致或者可以相互转换。但双精度小数型与小数型无须转换就可以直接进行比较。

7. “近似等于”命令：运算符号为“?”或“≈”

命令原型为：<逻辑型>近似等于(文本型 被比较文本, 文本型 比较文本)

“近似等于”命令是用于判断一个文本内是否包含另一个文本。如果是则返回“真”；否则返回“假”。

【注意】使用“近似等于”命令进行比较时是从两个文本的第一个字符开始一一对应比较的，如果用作比较的文本是被比较的文本中的一段，该命令返回的值依然是“假”。

例如：变量 b 为逻辑型，

b = “abcd” ≈ “bcd”

执行上述命令后变量 b 的值为“假”。

8. “并且”命令：运算符号为“&&”或“And”或“且”

命令原型为：<逻辑型>并且(逻辑型 逻辑值一, 逻辑型 逻辑值二, ...)

“并且”命令是用于判断两个或多个逻辑值是否都为“真”。如果是则返回“真”；否则返回“假”。

【注意】判断过程中如果其中一个参数的逻辑值为“假”，则命令立即返回，不会再继续判断下面的参数。此时，若作为参数的是一个子程序或命令的调用，那么它将不会被执行到。

9. “或者”命令：运算符号为“||”或“Or”或“或”

命令原型为：<逻辑型>或者(逻辑型 逻辑值一, 逻辑型 逻辑值二, ...)

“或者”命令是用于判断两个或多个逻辑值是否有一个为“真”。如果是则返回“真”；否则返回“假”。

【注意】判断过程中如果其中一个参数的逻辑值为“真”，则命令立即返回，不会再继续判断下面的参数。此时，若作为参数的是一个子程序或命令的调用，那么它将不会被执行到。

10. “取反”命令

命令原型为：<逻辑型>取反(逻辑型 被反转的逻辑值)

“取反”命令是用于获取一个逻辑值或逻辑型变量的值的相反值。如果被取反的值为“真”，则该命令返回“假”，否则返回“真”。

【注意】当一条逻辑判断条件中既使用了“或者”命令，同时又使用了“并且”命令，易语言中运算的顺序为：从左至右依次执行，并使用“或者”命令将多个条件分割，然后将使用“并且”命令连接的判断返回值看做成一个条件。最终整个判断的返回值是由“或者”命令返回的。可以理解为先处理“并且”，再处理“或者”，但是要注意程序并不是预先处理所有的“并且”命令再处理“或者”命令。

3.3.2 位运算

位运算是指对数据进行二进制的逐位运算。现代电子计算机内部是采用二进制方式存储

和处理数据的,输入到计算机的数字、字母、汉字等信息都以二进制的形式存储。在计算机内部都用“比特(bit)”这个单位来表示一个二进制,其状态只有0或1两种,位运算命令就是直接改变这些“比特”来达到改变数据的效果。

在易语言中,所有位运算命令都是针对“整数型”数据进行操作的,而“整数型”长度固定为4个字节 = 32 比特(位),也就是一组32位长度的二进制数。

1. “位取反”命令

命令原型为: <整数型> 位取反(整数型 欲取反的数值)

“位取反()”命令用于对一个数据的二进制值的每一位取反,即0变为1,1变为0,返回值的转换后的十进制数。

例如:编辑框_位取反.内容 = 到文本(位取反(60))

代码运行后,编辑框会显示“位取反”运算结果“-60”。

内部运算过程为:

- (1) 将60转换为二进制值,即0000 0000 0000 0000 0000 0000 0011 1100;
- (2) 将每位取反,即1111 1111 1111 1111 1111 1111 1100 0011;
- (3) 将取反结果转换为十进制整数型,即-60。

2. “位与”命令

命令原型为: <整数型> 位与(整数型 位运算数值一,整数型 位运算数值二,...)

“位与()”命令用于将2个或多个数据的二进制值中共同比特位进行“与”运算。即如两个或多个整数的共同位均为1,则返回值的对应位也为1,否则为0。运算后返回值是转换后的十进制数。

一个二进制数的第四位为1,另一个二进制数的第四位为1,则返回值的第四位为1;一个二进制数的第四位为0,另一个二进制数的第四位为1,则返回值的第四位为0;一个二进制数的第四位为1,另一个二进制数的第四位为0,则返回值的第四位为0;一个二进制数的第四位为0,另一个二进制数的第四位为0,则返回值的第四位为0。

例如:编辑框_位与.内容 = 到文本(位与(24,44))

运行后可以得出的结果为“8”。

24和44分别转换成二进制数为:0001 1000和0010 1100,进行与的运算后即会得出结果“0000 1000”即“8”。

3. “位或”命令

命令原型为: <整数型> 位或(整数型 位运算数值一,整数型 位运算数值二,...)

“位或()”命令对二进制值进行“或”运算,并将运算后结果以十进制返回。如果2个或多个数值的共同位均为0,则返回值的对应位也为0,否则为1。运算后返回值是转换后的十进制数。

一个数值的第四位为1,另一个数值的第四位为1,则返回值的第四位为1;一个数值的第四位为0,另一个数值的第四位为1,则返回值的第四位为1;一个数值的第四位为1,另一个数值的第四位为0,则返回值的第四位为1;一个数值的第四位为0,另一个数值的第四位为0,则返回值的第四位为0。

例如:编辑框_位或.内容 = 到文本(位或(24,44))

运行后的结果为“60”。

24 和 44 分别转换成二进制数为:0001 1000 和 0010 1100,进行或的运算后即会得出结果“0011 1100”即“60”。

4. “位异或”命令

命令原型为: <整数型>位异或(整数型 位运算数值一,整数型 位运算数值二,...)

“位异或()”命令对二进制数值的共同比特位进行“异或”运算,并将运算结果以十进制返回。如果两个或多个数值的共同位相等(均为 0 或均为 1),则返回值的对应位就是 0,否则为 1。运算结束返回值是转换后的十进制数。

一个数值的第四位为 0,另一个数值的第四位为 1,则返回值的第四位为 1;一个数值的第四位为 1,另一个数值的第四位为 0,则返回值的第四位为 1;一个数值的第四位为 1,另一个数值的第四位为 1,则返回值的第四位为 0;一个数值的第四位为 0,另一个数值的第四位为 0,则返回值的第四位为 0。

例如:编辑框_位异或.内容=到文本(位异或(24,44))

运行后的结果为“52”。

24 和 44 分别转换成二进制数为:0001 1000 和 0010 1100,进行异或的运算后即会得出结果“0011 0100”,即“52”。

5. “左移”命令

命令原型为: <整数型>左移(整数型 欲移动的整数,整数型 欲移动的位数)

“左移()”命令对二进制数据的每一位进行向左(高位)移动,移动后的空位自动用 0 补位。

参数 <2> 指定了欲移动的位数。如果该参数等于 0 或 32 的整数倍,则不进行移动操作并返回原值;如果该参数为正数,则现将该参数取 32 的余数,由右向左移动余数位;如果该参数为负数,则现将该参数取 32 的余数,并由右向左移动 32 + 余数位。

例如:左移(7,2)

将 7 转换成二进制数为 111(前 29 位为 0 占位),每一位向左移动 2 位,即 11100(前 27 位为 0 占位),再将结果以十进制返回,即 28。

例如:左移(7,31)

将 7 转换成二进制数为 111(前 29 位为 0 占位),每一位向左移动 31 位,即 111 后 31 个 0,但由于超过了 32 位,所以最前面的 11 将被抛弃,最终结果为 1 后面的 31 个 0,将结果以十进制返回,即 -2147483648。

6. “右移”命令

命令原型为: <整数型>右移(整数型 欲移动的整数,整数型 欲移动的位数)

“右移()”命令对二进制数据的每一位进行向右(低位)移动,移动后的空位自动用 0 补位。

参数 <2> 指定了欲移动的位数。如果该参数等于 0 或 32 的整数倍,则不进行移动操作并返回原值;如果该参数为正数,则先将该参数取 32 的余数,由左向右移动余数位;如果该参数为负数,则现将该参数取 32 的余数,并由左向右移动 32 + 余数位。

例如:右移(24,2)

将 24 转换成二进制数为 11000(前 27 位为 0 占位),每一位向右移动 2 位,即 110(前 29 位为 0 占位),将结果以十进制返回,即 6。

例如:右移(7,2)

将 7 转换成二进制数为 111(前 29 位为 0 占位),每一位向右移动 2 位,即前 31 个 0 + 111,但由于超过了 32 位,所以最后面的 11 将被抛弃,最终结果为前 31 个 0 + 1,再将结果以十进制返回,即 1。

7. “合并整数”命令

命令原型为: <整数型> 合并整数(整数型 用作合并的整数 1,整数型 用作合并的整数 2)

“合并整数()”命令是分别取两个整数的低 16 位并将他们组合成一个新的整数。从第一个参数中取得的 16 位将作为新整数的低 16 位;从第二个参数中取得的 16 位将作为新整数的高 16 位。

例如:合并整数(3,7)

将 2 的二进制数据中的后(右)16 位取出,即 14 个 0 + 11;

将 7 的二进制数据中的后(右)16 位取出,即 13 个 0 + 111;

将由 7 中取得的作为高位,由 3 中取得的作为低位进行组合。组合后的二进制结果为 13 个 0 + 111 + 14 个 0 + 11,并返回十进制结果,即 458755。

8. “合并短整数”命令

命令原型为: <短整数型> 合并短整数(整数型 用作合并的整数 1,整数型 用作合并的整数 2)

“合并短整数()”命令是分别取两个整数的低 8 位并将它们组合成一个新的短整数。从第一个参数中取得的 8 位将作为新整数的低 8 位;从第二个参数中取得的 16 位将作为新整数的高 16 位。

例如:合并短整数(3,7)

将 3 的二进制数据中的后(右)8 位取出,即 0000 0011;

将 7 的二进制数据中的后(右)8 位取出,即 0000 0111;

将由 7 中取得的作为高位,由 3 中取得的作为低位进行组合,组合后的二进制结果为 0000 0111 0000 0011,最后将结果以十进制返回,即 1795。

3.4 数组操作

数组,即相同数据类型变量的一个集合。一个数组的所有变量拥有同一个数组名,数组内的变量使用:数组名[下标]的方式表示数组内特定的一个变量,对数组中的特定成员的使用与该类型变量相同。

在程序中使用数组不但方便了管理相关的一组数据,而且方便了程序中使用循环控制一组数据。

数组在内存中是一段连续的数据。数组下标从 1 开始,数组可以定义为“多维”,多维数组可以看作每个成员都是一个数组的数组。数组成员数和对应维数可以动态地进行更改。

1. “重定义数组”命令

命令原型为: <无返回值> 重定义数组(通用型变量数组 欲重定义的数组变量,逻辑型 是否保留以前

的内容,整数型 数组对应维的上限值,...)

- (1) “重定义数组”命令用于在程序运行时动态地改变数组的结构。参数<2>指定了重定义后是否保留以前的内容。但如果重定义后,数组的对应维数小于原维数,则超出现维数的成员将被抛弃。
- (2) 【范例3-11】新建一个易语言 Windows 控制台程序,使用“重定义数组”命令改变具有6个成员的单维数组的结构,并在输出面板中显示转换结果。具体参考例程3-11。
- (3) 启动窗口程序集,如图3-21所示。
- (4) 【运行结果】运行例程3-11,观测输出面板中显示的内容,如图3-22所示。

子程序名	返回值类型	公开	备注
_启动子程序	整数型		本子程序在程序启动后最先执行

变量名	类型	静态	数组	备注
数组1	整数型		6	
变量1	整数型			

» 重定义数组 (数组1, 真, 5)
 变量1 = 取数组成员数 (数组1)
» 标准输出 (, 变量1)
 标准输入 (真)

图 3-21 “重定义数组”命令代码示例

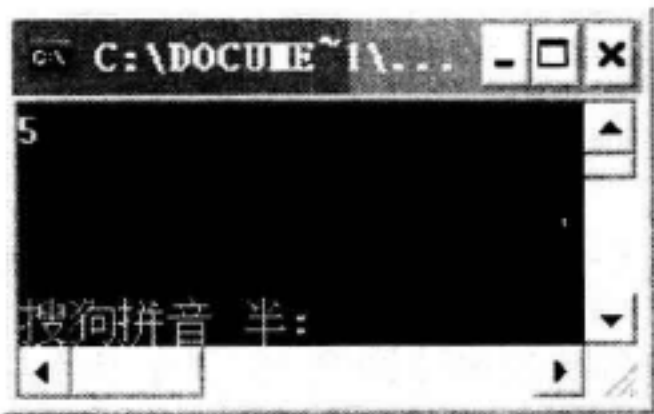


图 3-22 “重定义数组”命令运行结果示例

- (5) 【范例解析】原数组是一个有6个成员的单维数组,执行重定义数组命令后,现数组有5个成员,原数组中的第6个成员将被抛弃。

2. “取数组成员数”命令

命令原型为:<整数型>取数组成员数(通用型变量/变量数组 欲检查的变量)

- (1) “取数组成员数”命令用于获取一个数组中的成员数量。当被检查的数组是多维数组时,返回值为所有维成员的总和。
- (2) 【范例3-12】新建一个易语言 Windows 控制台程序,使用“取数组成员数”命令获取一个二维数组的成员数量,并在输出面板中显示结果。具体参考例程3-12。
- (3) 启动窗口程序集,如图3-23所示。
- (4) 【运行结果】运行例程3-12,观测输出面板中显示的内容,如图3-24所示。

子程序名	返回值类型	公开	备注
_启动子程序	整数型		本子程序在程序启动后最先执行

变量名	类型	静态	数组	备注
数组1	整数型		5, 6	
变量1	整数型			

变量1 = 到数值 (取数组成员数 (数组1))
» 标准输出 (, 变量1)
 标准输入 (真)

图 3-23 “取数组成员数”命令代码示例

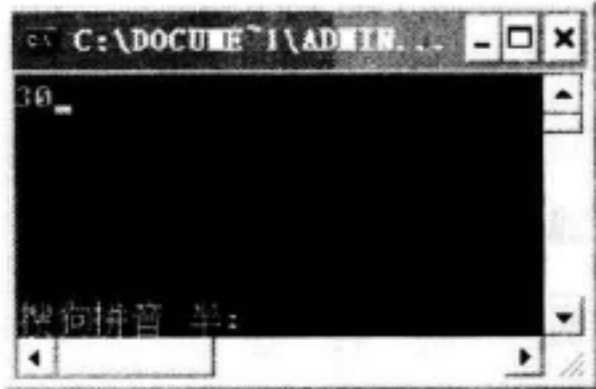


图 3-24 “取数组成员数”命令运行结果示例

- (5) 【范例解析】每一维具有6个成员的5维数组,其成员数为所有维成员的总和,即5×6,程序执行后,输出结果为30。

3. “取数组下标”命令

命令原型为:<整数型>取数组下标(通用型变量/变量数组 欲取某维最大下标的数组变量,[整数型 欲取其最大下标的维])

(1) “取数组下标”命令用于获取数组指定维的成员数。参数<2>指定了欲取数组中的第几维。如果给定的第一个参数不为数组变量或指定维不存在,则返回0。

(2) 【范例3-13】新建一个易语言 Windows 控制台程序,使用“取数组下标”命令获取一个五维数组第二维的成员数量,并在输出面板中显示结果。具体参考例程3-13。

(3) 启动窗口程序集,如图3-25所示。

(4) 【运行结果】运行例程3-13,观测输出面板中显示的内容,如图3-26所示。

子程序名	返回值类型	公开	备注
启动子程序	整数型		本子程序在程序启动后最先执行

变量名	类型	静态	数组	备注
数组1	整数型		5,6	
变量1	整数型			

变量1 = 到数值 (取数组下标 (数组1, 2))
 标准输出 (变量1)
 标准输入 (真)

图3-25 “取数组下标”命令代码示例

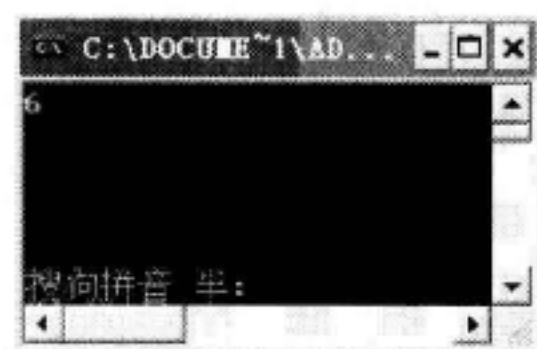


图3-26 “取数组下标”命令运行结果示例

(5) 【范例解析】该数组定义为每一维都具有6个成员的5维数组,其第二维的成员数量即为6,程序执行后,输出结果为6。

4. “复制数组”命令

命令原型为:<无返回值>复制数组(通用型变量数组 复制到的数组变量,通用型数组 待复制的数组数据)

(1) “复制数组”命令用于创建一个现有数组结构及内容的复制。目标数组的原结构和数据将被完全抛弃。

(2) 【范例3-14】新建一个易语言 Windows 控制台程序,使用“复制数组”命令将5维数组的结构及内容复制给目标数组,并获取目前数组第二维的成员数量,在输出面板中显示结果。具体参考例程3-14。

(3) 启动窗口程序集,如图3-27所示。

程序执行后,数组2的结构和数据与数组1完全相同。

输出目前数组第二维的最大下标,结果为6。

(4) 【运行结果】运行例程3-14,观测输出面板中显示的内容,如图3-28所示。

子程序名	返回值类型	公开	备注
启动子程序	整数型		本子程序在程序启动后最先执行

变量名	类型	静态	数组	备注
数组1	整数型		5,6	
数组2	整数型		0	
变量1	整数型			

复制数组 (数组2, 数组1)
 变量1 = 到数值 (取数组下标 (数组2, 2))
 标准输出 (变量1)
 标准输入 (真)

图3-27 “复制数组”命令代码示例

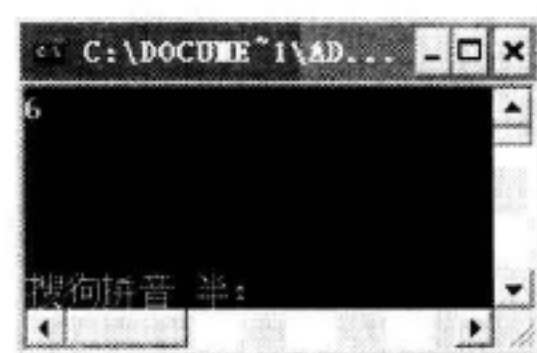


图3-28 “复制数组”命令运行结果示例

(5) 【范例解析】“复制数组”命令执行后,数组2的结构和数据与数组1完全相同,即每一维具有6个成员的5维数组,再通过执行“取数组下标”命令获取目前数组第二维的成员数,即输出该数组第二维的最大下标,结果为6。

5. “加入成员”命令

命令原型为：<无返回值>加入成员(通用型变量数组 欲加入成员的数组变量,通用型数组/非数组 欲加入的成员数据)

- (1) “加入成员”命令用于向现有数组中添加一个成员,添加的新成员在数组的最后。
- (2) 【范例 3-15】新建一个易语言 Windows 控制台程序,使用“加入成员”命令向 5 维数组中添加一个新成员,并获取目前数组第一维的成员数量,在输出面板中显示结果。具体参考例程 3-15。
- (3) 启动窗口程序集,如图 3-29 所示。
- (4) 【运行结果】运行例程 3-15,观测输出面板中显示的内容,如图 3-30 所示。

子程序名	返回值类型	公开	备注
启动子程序	整数型		本子程序在程序启动后最先执行

变量名	类型	静态	数组	备注
数组1	整数型		5	
变量1	整数型			

加入成员 (数组1, 123)
变量1 = 到数值 (取数组下标 (数组1, 1))
标准输出 (, 变量1)
标准输入 (真)

图 3-29 “加入成员”命令代码示例

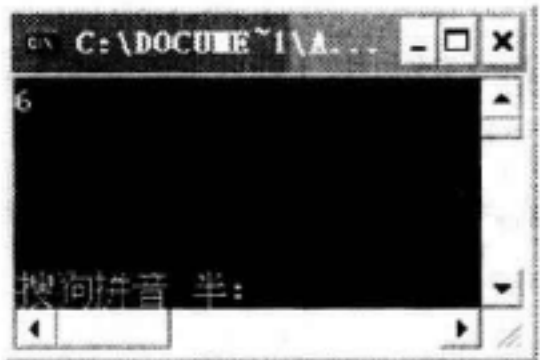


图 3-30 “加入成员”命令运行结果示例

- (5) 【范例解析】“加入成员”命令执行后,在数组 1 的最后添加了一个新成员,新成员的值 为 123,再通过执行“取数组下标”命令获取目前数组第一维的成员数,即输出该数组第一维 的最大下标,结果为 6。

6. “插入成员”命令

命令原型为：<无返回值>插入成员(通用型变量数组 欲插入成员的数组变量,整数型 欲插入的位置, 通用型数组/非数组 欲插入的成员数据)

- (1) “插入成员”命令用于向现有的数组中指定的位置插入一个成员。参数 <2> 指定了 欲插入成员的位置。位置值从 1 开始,如果小于 1 或大于现数组的成员数目 + 1,将不会插入 任何数据。
- (2) 【范例 3-16】新建一个易语言 Windows 控制台程序,使用“插入成员”命令向具有 5 个成员的 单维数组中插入一个新成员,并通过输出面板输出当前数组中第 3 个成员的值。具体参考 例程 3-16。
- (3) 启动窗口程序集,如图 3-31 所示。
- (4) 【运行结果】运行例程 3-16,观测输出面板中显示的内容,如图 3-32 所示。

子程序名	返回值类型	公开	备注
启动子程序	整数型		本子程序在程序启动后最先执行

变量名	类型	静态	数组	备注
数组1	整数型		5	
变量1	整数型			

插入成员 (数组1, 3, 123)
变量1 = 数组1 [3]
标准输出 (, 变量1)
标准输入 (真)

图 3-31 “插入成员”命令代码示例

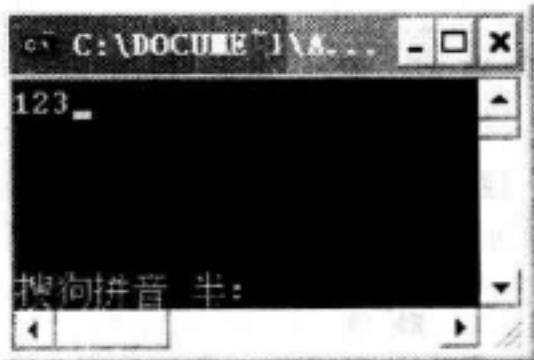


图 3-32 “插入成员”命令运行结果示例

(5) 【范例解析】“插入成员”命令执行后,在单维数组的第3个成员前插入了一个新成员,所以当前数组中第3个成员的值即为新成员的值123。

7. “删除成员”命令

命令原型为: <无返回值> 删除成员(通用型变量数组 欲删除成员的数组变量, 整数型 欲删除的位置, [整数型 欲删除的成员数目])

(1) “删除成员”命令用于在现有数组中删除指定的成员。数组变量如果为多维数组,删除完毕后将转换为单维数组,命令执行后返回所实际删除的成员数目。

(2) 【范例3-17】新建一个易语言 Windows 控制台程序,使用“删除成员”命令将多维数组中的某些成员删除,并在输出面板输出实际被删除的成员数量及数组中现有成员的数量。具体参考例程3-17。

(3) 启动窗口程序集,如图3-33所示。

(4) 【运行结果】运行例程3-17,观测输出面板中显示的内容,如图3-34所示。

子程序名	返回值类型	公开	备注
启动子程序	整数型		本子程序在程序启动后最先执行

变量名	类型	静态	数组	备注
数组1	整数型		5,6	
变量1	整数型			
变量2	整数型			

变量1 = 到数值 (删除成员 (数组1, 10, 30))
 变量2 = 取数组成员数 (数组1)
 标准输出 (到文本 (变量1) + “,” + 到文本 (变量2))
 标准输入 (真)

图 3-33 “删除成员”命令代码示例



图 3-34 “删除成员”命令运行结果示例

(5) 【范例解析】程序执行后,数组1将被转换为单维数组。因为原数组共有30个成员,从第10个(包括第10个)向后删除30个成员,所以实际被删除的只有下标为10到30的成员。在输出面板中程序输出了实际被删除的成员数量21,和数组现有成员的数量9。

8. “清除数组”命令

命令原型为: <无返回值> 清除数组(通用型变量数组 欲删除成员的数组变量)

(1) “清除数组”命令用于删除指定数组变量中的所有成员,释放这些成员所占用的存储空间,重新定义该变量为单维0成员数组变量。

(2) 【范例3-18】新建一个易语言 Windows 控制台程序,使用“清除数组”命令将多维数组中的所有成员删除,并在输出面板输出数组中现有成员的数量。具体参考例程3-18。

(3) 启动窗口程序集,如图3-35所示。

(4) 【运行结果】运行例程3-18,观测输出面板中显示的内容,如图3-36所示。

子程序名	返回值类型	公开	备注
启动子程序	整数型		本子程序在程序启动后最先执行

变量名	类型	静态	数组	备注
数组1	整数型		5,6	
变量1	整数型			

清除数组 (数组1)
 变量1 = 取数组成员数 (数组1)
 标准输出 (变量1)
 标准输入 (真)

图 3-35 “清除数组”命令代码示例

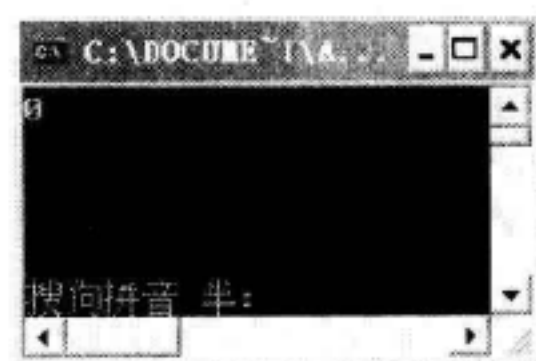


图 3-36 “清除数组”命令运行结果示例

(5) 【范例解析】程序执行后,数组 1 中的所有成员被删除,该数组即为单维 0 成员数组,所以其成员数量为 0。

9. “数组排序”命令

命令原型为: <无返回值>数组排序(通用型变量数组 数值数组变量,[逻辑型 排序方向是否为从小到大])

(1) “数组排序”命令用于对一个数值型数组进行有序的排列。

参数 <2> 指定了排序的方式,如果参数被省略则默认使用从小到大排序。如果被排序的数组是多维数组,排序后的结果按照先低后高的维数进行排列。

例如:排序一个二维数组,两个维数下标上限都为 2,排序后的第一个结果将存放在[1][1]内,第二个结果将存放在[1][2]内,第三个结果将存放在[1][3]内,其他多维数的数组依此类推。

(2) 【范例 3-19】新建一个易语言 Windows 控制台程序,使用“数组排序”命令对多维数组进行排序,并在输出面板输出数组中所有成员的最小值。具体参考例程 3-19。

(3) 启动窗口程序集,如图 3-37 所示。

(4) 【运行结果】运行例程 3-19,观测输出面板中显示的内容,如图 3-38 所示。

子程序名	返回值类型	公开	备注
启动子程序	整数型		本子程序在程序启动后最先执行

变量名	类型	静态	数组	备注
数组1	整数型		2, 2	
变量1	整数型			

```

数组1 [1] [1] = 3
数组1 [2] [1] = 2
数组排序 (数组1, )
变量1 = 数组1 [1] [1]
标准输出 ( 变量1)
标准输入 (真)

```

图 3-37 “数组排序”命令代码示例



图 3-38 “数组排序”命令运行结果示例

(5) 【范例解析】程序运行后,首先对数组的[1][1]和[2][1]赋值,其次执行“数组排序”命令,数组将按照从小到大顺序排列,最后输出数组中[1][1]成员的值,即最小值,显示结果如图 3-38 所示。

10. “数组清零”命令

命令原型为: <无返回值>数组清零(通用型变量数组 数值数组变量)

(1) “数组清零”命令用于对数值型数组的每个成员进行置 0 操作。

(2) 【范例 3-20】新建一个易语言 Windows 控制台程序,使用“数组清零”命令对多维数组进行操作,并在输出面板输出数组中第一个成员的值。具体参考例程 3-20。

(3) 启动窗口程序集,如图 3-39 所示。

(4) 【运行结果】运行例程 3-20,观测输出面板中显示的内容,如图 3-40 所示。

子程序名	返回值类型	公开	备注
启动子程序	整数型		本子程序在程序启动后最先执行

变量名	类型	静态	数组	备注
数组1	整数型		2, 2	
变量1	整数型			

```

数组1 [1] [1] = 3
数组清零 (数组1)
变量1 = 数组1 [1] [1]
标准输出 ( 变量1)
标准输入 (真)

```

图 3-39 “数组清零”命令代码示例

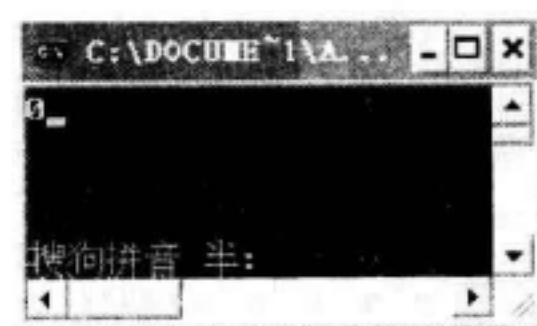


图 3-40 “数组清零”命令运行结果示例

(5) 【范例解析】执行数组清零命令后,数组中的每个成员的值都为 0,最后输出[1][1]成员的值,即为 0。

3.5 数值转换

1. “到数值”命令

命令原型为: <双精度小数型> 到数值(通用型 待转换的文本或数值)

- (1) “到数值”命令用于将其他类型数据转换为“双精度小数型”数据。
- (2) 【范例 3-21】新建一个 Windows 控制台程序,通过使用“到数值”命令将文本型数据 612.47 转换为双精度型,并在输出面板中显示转换结果。具体参考例程 3-21。
- (3) 启动窗口程序集,如图 3-41 所示。
- (4) 【运行结果】运行例程 3-21,观测输出面板中显示的内容,如图 3-42 所示。

子程序名	返回值类型	公开	备注
_启动子程序	整数型		

变量名	类型	静态	数组	备注
变量1	双精度小数型			

变量1 = 到数值 (“612.47”)
标准输出 (变量1)
标准输入 0

图 3-41 “到数值”命令代码示例

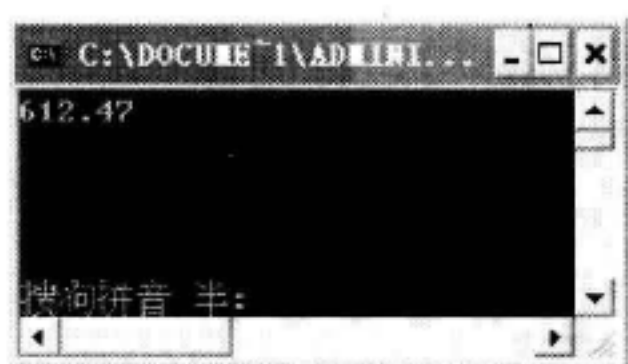


图 3-42 “到数值”命令运行结果示例

(5) 【范例解析】程序运行后文本型“612.47”将被转换为双精度小数型数据,并保存到“变量 1”中,并在输出面板中显示“变量 1”的值,结果为“612.47”。

2. “数值到大写”命令

命令原型为: <文本型> 数值到大写(双精度小数型 欲转换形式的数值,逻辑型 是否转换为简体)

- (1) “数值到大写”命令用于将双精度小数型数据转换为中文大写文本。参数 <2> 指定了返回时文本的样式。若为“真”,则返回表示数值的汉字类型;若为“假”,则返回表示金额的汉字类型。

【注意】小数部分只支持两位小数。

(2) 【范例 3-22】新建一个 Windows 控制台程序,通过使用“数值到大写”命令将双精度型 612.47 转换为中文大写的文本格式,并在输出面板中显示转换结果。具体参考例程 3-22。

(3) 启动窗口程序集,如图 3-43 所示。

(4) 【运行结果】运行例程 3-22,观测输出面板中显示的内容,如图 3-44 所示。

子程序名	返回值类型	公开	备注
_启动子程序	整数型		

变量名	类型	静态	数组	备注
变量1	文本型			

变量1 = 数值到大写 (612.47, 假)
标准输出 (变量1)
标准输入 (真)

图 3-43 “数值到大写”命令代码示例

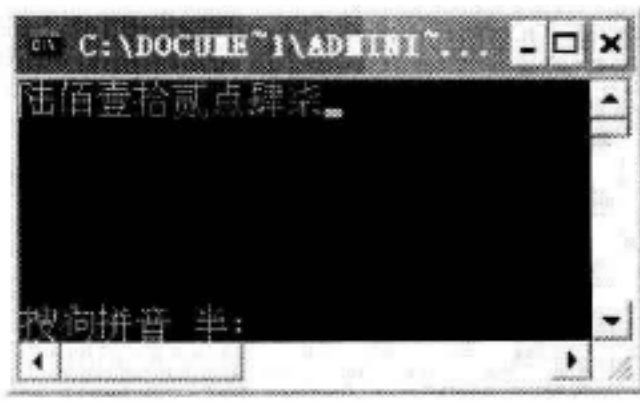


图 3-44 “数值到大写”命令运行结果示例

(5) 【范例解析】程序运行后将“612.47”转换为文本型汉字的表示方式,并保存到“变量1”中,并在输出面板中显示“变量1”的值,显示结果为“陆佰壹拾贰点肆柒”。

3. “数值到金额”命令

命令原型为: <文本型> 数值到金额(双精度小数型 欲转换形式的数值,逻辑型 是否转换为简体)

(1) “数值到金额”命令用于将双精度小数型数据转换为以人民币作为单位的汉字书写形式。

【注意】该命令在处理过程中只处理到百分位,千分位将被四舍五入。

(2) 【范例3-23】新建一个 Windows 控制台程序,通过使用“数值到金额”命令将双精度型 425.7887 转换为以人民币作为单位的汉字书写格式,并在输出面板中显示转换结果。具体参考例程 3-23。

(3) 启动窗口程序集,如图 3-45 所示。

(4) 【运行结果】运行例程 3-23,观测输出面板中显示的内容,如图 3-46 所示。

子程序名	返回值类型	公开	备注
启动子程序	整数型		

变量名	类型	静态	数组	备注
变量1	文本型			

变量1 = 数值到金额 (425.7887, 假)
 标准输出 (变量1)
 标准输入 (真)

图 3-45 “数值到金额”命令代码示例

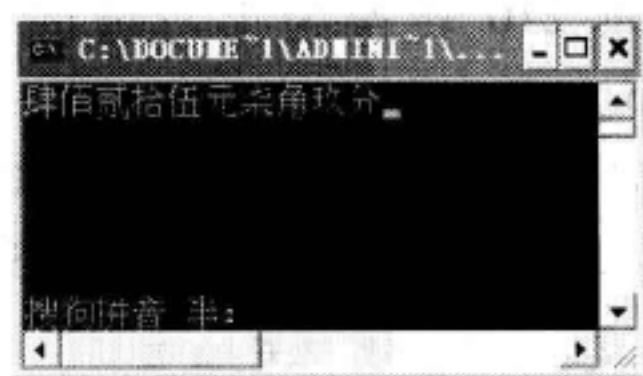


图 3-46 “数值到金额”命令运行结果示例

(5) 【范例解析】程序运行后将“425.7887”转换为人民币汉字书写方式的文本,并保存到“变量1”中,并在输出面板中显示“变量1”的值,显示结果为“肆佰贰拾伍元柒角玖分”。

4. “数值到格式文本”命令

命令原型为: <文本型> 数值到格式文本(双精度小数型 欲转换为文本的数值,[整数型 小数保留位数],逻辑型 是否进行千分位分隔)

(1) “数值到格式文本”命令用于将双精度小数型数据转换为指定格式的文本。

参数 <2> 指定了转换时保留小数的位数。如果大于 0,表示小数点右边应四舍五入保留的位数;如果等于 0,表示舍入到整数;如果小于 0,表示小数点左边舍入到的位置。例如:数值到格式文本(2163.58, 1, 假)返回值为“2163.6”;数值到格式文本(2163.58, 0, 假)返回“2164”;数值到格式文本(2163.58, -1, 假)则返回“2160”。如果省略本参数,则默认为保留所有实际存在的小数位。

参数 <3> 指定了是否进行每三位整数(从右向左数)使用逗号进行分隔。若为“真”则使用,否则不使用。

(2) 【范例3-24】新建一个 Windows 控制台程序,通过使用“数值到格式文本”命令将双精度型数据 3942.6512 转换为文本型数据,要求默认保留所有小数位,同时进行千分位分隔,并在输出面板中显示转换结果。具体参考例程 3-24。

(3) 启动窗口程序集,如图 3-47 所示。

(4) 【运行结果】运行例程 3-24,观测输出面板中显示的内容,如图 3-48 所示。

(5) 【范例解析】程序运行后将“3942.6512”转换为文本型数据,并为每 3 位整数添加一个逗号做分隔,将转换的结果保存在“变量1”中,并在输出面板中显示“变量1”的内容,显示

结果为“3,942.6512”。

子程序名	返回值类型	公开	备注
启动子程序	整数型		

变量名	类型	静态	数组	备注
变量1	文本型			

变量1 = 数值到格式文本 (3942.6512, , 真)
标准输出 (, 变量1)
标准输入 (真)

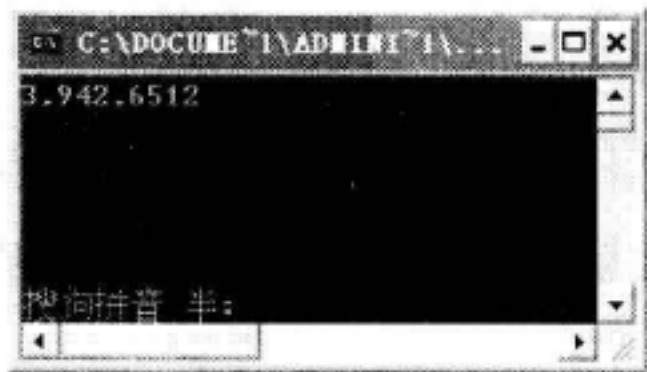


图 3-47 “数值到格式文本”命令代码示例 图 3-48 “数值到格式文本”命令运行结果示例

5. “取十六进制文本”命令

命令原型为：<文本型>取十六进制文本(整数型 欲取进制文本的数值)

(1) “取十六进制文本”命令用于将一个整数型数据转换为十六进制数的文本。

(2) 【范例 3-25】新建一个 Windows 控制台程序,通过使用“取十六进制文本”命令将十进制数文本 210 转换为十六进制数文本,并在输出面板中显示转换结果。具体参考例程 3-25。

(3) 启动窗口程序集,如图 3-49 所示。

(4) 【运行结果】运行例程 3-25,观测输出面板中显示的内容,如图 3-50 所示。

子程序名	返回值类型	公开	备注
启动子程序	整数型		

变量名	类型	静态	数组	备注
变量1	文本型			

变量1 = 取十六进制文本 (210)
标准输出 (, 变量1)
标准输入 (真)

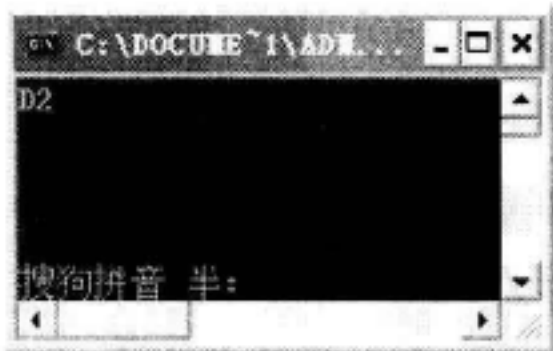


图 3-49 “取十六进制文本”命令代码示例 图 3-50 “取十六进制文本”命令运行结果示例

(5) 【范例解析】程序运行后将十进制数文本“210”转换为十六进制数文本,将转换的结果保存在“变量 1”中,并在输出面板中显示“变量 1”的内容,显示结果为“D2”。

6. “取八进制文本”命令

命令原型为：<文本型>取八进制文本(整数型 欲取进制文本的数值)

(1) “取八进制文本”命令用于将一个整数型数据转换为八进制数的文本。

(2) 【范例 3-26】新建一个 Windows 控制台程序,通过使用“取八进制文本”命令将十进制数文本 210 转换为八进制数文本,并在输出面板中显示转换结果。具体参考例程 3-26。

(3) 启动窗口程序集,如图 3-51 所示。

(4) 【运行结果】运行例程 3-26,观测输出面板中显示的内容,如图 3-52 所示。

子程序名	返回值类型	公开	备注
启动子程序	整数型		

变量名	类型	静态	数组	备注
变量1	文本型			

变量1 = 取八进制文本 (210)
标准输出 (, 变量1)
标准输入 (真)



图 3-51 “取八进制文本”命令代码示例 图 3-52 “取八进制文本”命令运行结果示例

(5) 【范例解析】程序运行后将十进制数文本“210”转换为八进制数文本,将转换的结果保存在“变量1”中,并在输出面板中显示“变量1”的内容,显示结果为“322”。

7. “到字节”命令

命令原型为: <字节型>到字节(通用型 待转换的文本或数值)

(1) “到字节”命令用于将其他数据类型的数据转换为字节型数据。转换的过程中如果有小数,小数部分将被抛弃。该命令支持转换全角文字类型的数字。

(2) 【范例3-27】新建一个 Windows 控制台程序,通过使用“到字节”命令将文本型数据 63.48 转换为字节型数据,并在输出面板中显示转换结果。具体参考例程 3-27。

(3) 启动窗口程序集,如图 3-53 所示。

(4) 【运行结果】运行例程 3-27,观测输出面板中显示的内容,如图 3-54 所示。

子程序名	返回值类型	公开	备注
启动子程序	整数型		

变量名	类型	静态	数组	备注
变量1	字节型			

变量1 = 到字节 (63.48)
 标准输出 (变量1)
 标准输入 (真)

图 3-53 “到字节”命令代码示例

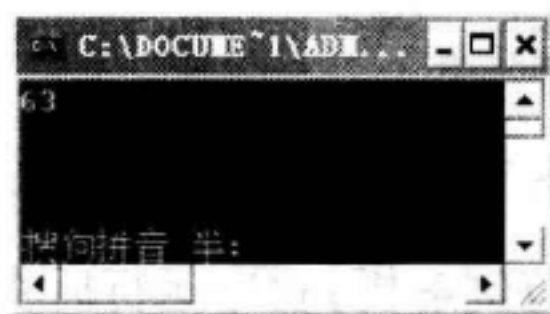


图 3-54 “到字节”命令运行结果示例

(5) 【范例解析】程序运行后将文本型“63.48”转换为字节型,由于有小数,所以小数部分被完全抛弃,只保留了整数部分,将转换的结果保存在“变量1”中,并在输出面板中显示“变量1”的内容,显示结果为“63”。

8. “到短整数”命令

命令原型为: <短整数型>到短整数(通用型 待转换的文本或数值)

(1) “到短整数”命令用于将其他数据类型的数据转换为短整数型数据。转换的过程中如果有小数,小数部分将被抛弃。该命令同样支持转换全角文字类型的数字。

(2) 【范例3-28】新建一个 Windows 控制台程序,通过使用“到短整数”命令将文本型数据 310.94 转换为短整数型数据,并在输出面板中显示转换结果。具体参考例程 3-28。

(3) 启动窗口程序集,如图 3-55 所示。

(4) 【运行结果】运行例程 3-28,观测输出面板中显示的内容,如图 3-56 所示。

子程序名	返回值类型	公开	备注
启动子程序	整数型		

变量名	类型	静态	数组	备注
变量1	短整数型			

变量1 = 到短整数 (“310.94”)
 标准输出 (变量1)
 标准输入 (真)

图 3-55 “到短整数”命令代码示例

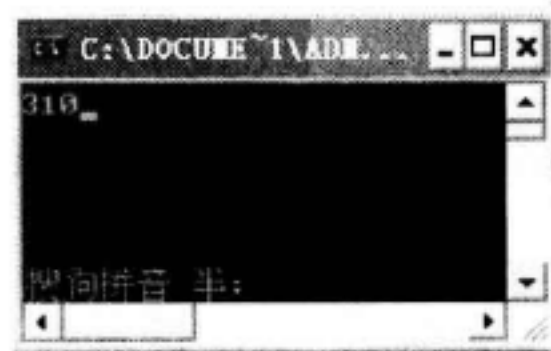


图 3-56 “到短整数”命令运行结果示例

(5) 【范例解析】程序运行后将文本型“310.94”转换为短整数型“310”,小数部分被完全抛弃,只保留了整数部分,将转换的结果保存在“变量1”中,并在输出面板中显示“变量1”的内容,显示结果为“310”。

9. “到整数”命令

命令原型为：<整数型>到整数(通用型 待转换的文本或数值)

(1) “到整数”命令用于将其他数据类型的数据转换为整数型数据。转换的过程中如果有小数,小数部分将被抛弃。该命令同样支持转换全角文字类型的数字。

(2) 【范例 3-29】新建一个 Windows 控制台程序,通过使用“到整数”命令将文本型数据 235.41 转换为整数型数据,并在输出面板中显示转换结果。具体参考例程 3-29。

(3) 启动窗口程序集,如图 3-57 所示。

(4) 【运行结果】运行例程 3-29,观测输出面板中显示的内容,如图 3-58 所示。

子程序名	返回值类型	公开	备注
启动子程序	整数型		

变量名	类型	静态	数组	备注
变量1	整数型			

变量1 = 到整数 (“235.41”)
 标准输出 (变量1)
 标准输入 (真)

图 3-57 “到整数”命令代码示例

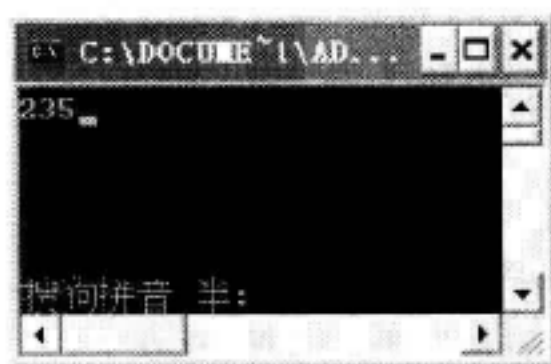


图 3-58 “到整数”命令运行结果示例

(5) 【范例解析】程序运行后将文本型“235.41”转换为整数型“235”,小数部分被完全抛弃,只保留了整数部分,将转换的结果保存在“变量 1”中,并在输出面板中显示“变量 1”的内容,显示结果为“235”。

10. “到长整数”命令

命令原型为：<长整数型>到长整数(通用型 待转换的文本或数值)

(1) “到长整数”命令用于将其他数据类型的数据转换为长整数型数据。转换的过程中如果有小数,小数部分将被抛弃。该命令同样支持转换全角文字类型的数字。

(2) 【范例 3-30】新建一个 Windows 控制台程序,通过使用“到长整数”命令将文本型数据 75.68 转换为长整数型数据,并在输出面板中显示转换结果。具体参考例程 3-30。

(3) 启动窗口程序集(图 3-59)。

(4) 【运行结果】运行例程 3-30,观测输出面板中显示的内容,如图 3-60 所示。

子程序名	返回值类型	公开	备注
启动子程序	整数型		

变量名	类型	静态	数组	备注
变量1	长整数型			

变量1 = 到长整数 (“75.68”)
 标准输出 (变量1)
 标准输入 (真)

图 3-59 “到长整数”命令代码示例

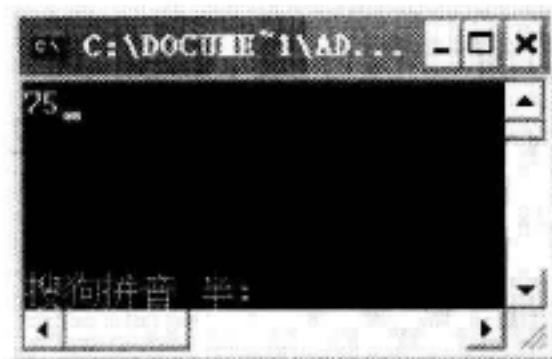


图 3-60 “到长整数”命令运行结果示例

(5) 【范例解析】程序运行后将文本型“75.68”转换为长整数型“75”,小数部分被完全抛弃,只保留了整数部分,将转换的结果保存在“变量 1”中,并在输出面板中显示“变量 1”的内容,显示结果为“75”。

11. “到小数”命令

命令原型为：<小数型>到小数(通用型 待转换的文本或数值)

(1) “到小数”命令用于将其他数据类型的数据转换为小数型数据。该命令同样支持转

换全角文字类型的数字。

(2) 【范例 3-31】新建一个 Windows 控制台程序,通过使用“到小数”命令将全角的文本型数据 618.7 转换为小数型数据,并在输出面板中显示转换结果。具体参考例程 3-31。

(3) 启动窗口程序集(图 3-61)。

(4) 【运行结果】运行例程 3-31,观测输出面板中显示的内容,如图 3-62 所示。

子程序名	返回值类型	公开	备注
启动子程序	整数型		

变量名	类型	静态	数组	备注
变量1	小数型			

变量1 = 到小数 (“6 1 8 . 7 ”)

标准输出 (变量1)

标准输入 (真)

图 3-61 “到小数”命令代码示例

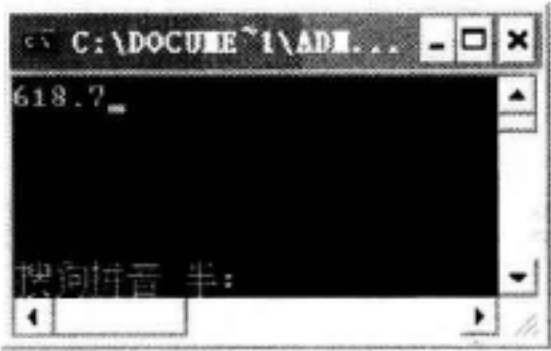


图 3-62 “到小数”命令运行结果示例

(5) 【范例解析】程序运行后将全角的文本型“618.7”转换为小数型数据,转换的结果保存在“变量 1”中,并在输出面板中显示“变量 1”的内容,显示结果为“618.7”。

第4章 易语言常用命令

4.1 时间操作

时间操作类命令也是较常用的一类命令,可以对日期时间型数据进行操作。

1. 到时间

调用格式:〈日期时间型〉到时间(欲转换的文本)

英文名称:ToTime

将指定文本转换为时间并返回。如果给定文本不符合书写格式要求或者时间值错误导致不能进行转换,将返回100年1月1日。如果给定参数本身就是时间数据,将直接返回该时间。本命令为初级命令。

参数<1>的名称为“欲转换的文本”,类型为“通用型(all)”。文本内容应按以下格式之一提供,年份后的时间部分可以省略:

(1) 1973年11月15日12时30分25秒

(2) 1973/11/15 12:30:25

(3) 1973/11/15/12/30/25

(4) 1973/11/15/12:30:25

(5) 1973-11-15-12-30-25

(6) 1973-11-15-12:30:25

(7) 19731115123025

2. 增减时间

调用格式:〈日期时间型〉增减时间(时间,被增加部分,增加值)

英文名称:TimeChg

返回一个时间,这一时间被加上或减去了一段间隔。如果改变后的时间无效,将自动靠近最近的有效时间。本命令为初级命令。

参数<1>的名称为“时间”,类型为“日期时间型(date)”。

参数<2>的名称为“被增加部分”,类型为“整数型(int)”。参数值指定增加或减少时间的哪一部分,可以为以下常量:(1)#年份;(2)#季度;(3)#月份;(4)#周;(5)#日;(6)#小时;(7)#分钟;(8)#秒。

参数<3>的名称为“增加值”,类型为“整数型(int)”。相对于被增加部分的增加或减少数值。

3. 取时间间隔

调用格式:〈双精度小数型〉取时间间隔(时间1,时间2,取间隔部分)

英文名称:TimeDiff

返回一个数值,表示“时间1”减去“时间2”之后的间隔数目。注意:每个星期以星期天为第一天。本命令为初级命令。

参数<1>的名称为“时间1”,类型为“日期时间型(date)”。

参数<2>的名称为“时间2”,类型为“日期时间型(date)”。

参数<3>的名称为“取间隔部分”,类型为“整数型(int)”。参数值指定取时间间隔的哪一部分,可以为以下常量:(1)#年份;(2)#季度;(3)#月份;(4)#周;(5)#日;(6)#小时;(7)#分钟;(8)#秒。

4. 取某月天数

调用格式:〈整数型〉取某月天数(年份,月份)

英文名称:GetDaysOfSpecMonth

返回指定月份所包含的天数。如果给定的月份无效,返回0。本命令为初级命令。

参数<1>的名称为“年份”,类型为“整数型(int)”。参数值从100到9999。

参数<2>的名称为“月份”,类型为“整数型(int)”。参数值从1到12。

5. 时间到文本

调用格式:〈文本型〉时间到文本(欲转换到文本的时间,[转换部分])

英文名称:TimeToText

将指定时间转换为文本并返回。本命令为初级命令。

参数<1>的名称为“欲转换到文本的时间”,类型为“日期时间型(date)”。

参数<2>的名称为“转换部分”,类型为“整数型(int)”,可以被省略。参数值可以为以下常量:(1)#全部转换;(2)#日期部分;(3)#时间部分。如果省略了本参数,默认为“#全部转换”。

6. 取时间部分

调用格式:〈整数型〉取时间部分(欲取其部分的时间,欲取的时间部分)

英文名称:TimePart

返回一个包含已知时间指定部分的整数。本命令为初级命令。

参数<1>的名称为“欲取其部分的时间”,类型为“日期时间型(date)”。

参数<2>的名称为“欲取的时间部分”,类型为“整数型(int)”。参数值可以为以下常量:(1)#年份;(2)#季度;(3)#月份;(4)#自年首周数;(5)#日;(6)#小时;(7)#分钟;(8)#秒;(9)#星期几;(10)#自年首天数。其中:自年首周数、自年首天数均从1开始。

7. 取年份

调用格式:〈整数型〉取年份(时间)

英文名称:year

返回一个值为100到9999之间的整数,表示指定时间中的年份。本命令为初级命令。

参数<1>的名称为“时间”,类型为“日期时间型(date)”。

8. 取月份

调用格式:〈整数型〉取月份(时间)

英文名称:month

返回一个值为1到12之间的整数,表示指定时间中的月份。本命令为初级命令。

参数<1>的名称为“时间”,类型为“日期时间型(date)”。

9. 取日

调用格式:〈整数型〉取日(时间)

英文名称:day

返回一个值为1到31之间的整数,表示一个月中的某一日。本命令为初级命令。

参数<1>的名称为“时间”,类型为“日期时间型(date)”。

10. 取星期几

调用格式:〈整数型〉取星期几(时间)

英文名称:WeekDay

返回一个值为 1 到 7 之间的整数,表示一个星期中的某一日。星期日为 1,星期一为 2,依此类推。本命令为初级命令。

参数<1>的名称为“时间”,类型为“日期时间型(date)”。

11. 取小时

调用格式:〈整数型〉取小时(时间)

英文名称:hour

返回一个值为 0 到 23 之间的整数,表示一天中的某一小时。本命令为初级命令。

参数<1>的名称为“时间”,类型为“日期时间型(date)”。

12. 取分钟

调用格式:〈整数型〉取分钟(时间)

英文名称:minute

返回一个值为 0 到 59 之间的整数,表示一小时中的某一分钟。本命令为初级命令。

参数<1>的名称为“时间”,类型为“日期时间型(date)”。

13. 取秒

调用格式:〈整数型〉取秒(时间)

英文名称:second

返回一个值为 0 到 59 之间的整数,表示一分钟中的某一秒。本命令为初级命令。

参数<1>的名称为“时间”,类型为“日期时间型(date)”。

14. 指定时间

调用格式:〈日期时间型〉指定时间(年,[月],[日],[小时],[分钟],[秒])

英文名称:GetSpecTime

返回包含指定年、月、日、小时、分、秒的时间。如果指定了无效时间,将自动使用最相近的有效时间代替。本命令为初级命令。

参数<1>的名称为“年”,类型为“整数型(int)”。

参数<2>的名称为“月”,类型为“整数型(int)”,可以被省略。如果本参数被省略,默认为 1 月。

参数<3>的名称为“日”,类型为“整数型(int)”,可以被省略。如果本参数被省略,默认为 1 日。

参数<4>的名称为“小时”,类型为“整数型(int)”,可以被省略。如果本参数被省略,默认为 0 时。

参数<5>的名称为“分钟”,类型为“整数型(int)”,可以被省略。如果本参数被省略,默认为 0 分钟。

参数<6>的名称为“秒”,类型为“整数型(int)”,可以被省略。如果本参数被省略,默认为 0 秒。

15. 取现行时间

调用格式:〈日期时间型〉取现行时间()

英文名称:now

返回当前系统日期及时间。本命令为初级命令。

16. 置现行时间

调用格式:〈逻辑型〉置现行时间(欲设置的时间)

英文名称:SetSysTime

设置当前系统日期及时间。成功返回真,失败返回假。本命令为初级命令。

参数<1>的名称为“欲设置的时间”,类型为“日期时间型(date)”。

17. 取日期

调用格式:〈日期时间型〉取日期(时间)

英文名称:GetDatePart

返回一个日期时间型数据的日期部分,其小时、分钟、秒被固定设置为0时0分0秒。本命令为初级命令。

参数<1>的名称为“时间”,类型为“日期时间型(date)”。

18. 取时间

调用格式:〈日期时间型〉取时间(时间)

英文名称:GetTimePart

返回一个日期时间型数据的时间部分,其年、月、日被固定设置为2000年1月1日。本命令为初级命令。

参数<1>的名称为“时间”,类型为“日期时间型(date)”。

4.2 文本操作与字节集操作

在学习易语言文本操作类命令之前,先来了解一下计算机处理文字的方式和相关的概念。

4.2.1 文字编码和存储方式

在计算机中,所有的数据存储和运算时都要使用二进制数表示(计算机用高电平和低电平分别表示1和0)。例如:像a、b、c、d……这样的52个字母(包括大小写)以及0、1、2、3、4、5、6、7、8、9数字还有一些常用的符号(*、#、@等)在计算机中存储时也要使用二进制数来表示,而具体用哪些二进制数字表示哪个符号,当然每个人都可以约定自己的一套(这就叫编码),而相互之间既要相互通信又不造成混乱,那么就必须使用相同的编码规则,于是美国有关的标准化组织就出台了所谓的ASCII编码,统一规定了上述常用符号用哪些二进制数来表示。

【提示】除了ASCII码,常用的编码还有Unicode码等字符编码。每套编码集中编码值所对应的字符不尽相同。

4.2.2 ASCII码

ASCII(American Standard Code for Information Interchange,美国信息互换标准代码)是基于拉丁字母的一套计算机编码系统。它主要用于显示现代英语和其他西欧语言。它是现今最通用的单字节编码系统,并等同于国际标准ISO/IEC646。

ASCII码使用指定的7位或8位二进制数组来表示128或256种可能的字符。标准ASCII码也叫基础ASCII码,使用7位二进制数来表示所有的大写字母和小写字母,数字0到9、标点

符号以及在美式英语中使用的特殊控制字符。其中:

0~31 及 127(共 33 个)是控制字符或通信专用字符(其余为可显示字符),如控制符:LF(换行)、CR(回车)、FF(换页)、DEL(删除)、BS(退格)、BEL(振铃)等;通信专用字符:SOH(文头)、EOT(文尾)、ACK(确认)等;ASCII 值为 8、9、10、13 分别转换为退格、制表、换行、回车符号。它们并没有特定的图形显示,但会依不同的应用程序,而对文本显示有不同的影响。

32~126(共 95 个)是字符(32 是空格),其中 48~57 为 0 到 9 这 10 个阿拉伯数字;65~90 为 26 个大写英文字母,97~122 为 26 个小写英文字母,其余为一些标点符号、运算符号等。

在标准 ASCII 中,最高位(b7)用作奇偶校验位。所谓奇偶校验,是指在代码传送过程中用来检验是否出现错误的一种方法,一般分为奇校验和偶校验两种。奇校验规定:正确的代码一个字节中 1 的个数必须是奇数,若非奇数,则在最高位 b7 添 1;偶校验规定:正确的代码一个字节中 1 的个数必须是偶数,若非偶数,则在最高位 b7 添 0。

后 128 个称为扩展 ASCII 码,目前许多基于 X86 的系统都支持使用扩展(或“高位”)ASCII。扩展 ASCII 码允许将每个字符的第 8 位用于确定附加 128 位特殊符号字符、外来与字母和图形符号。

注解:

(1) 易语言中使用的是十进制 ASCII 码值。

(2) 常见 ASCII 码值的大小规律:

a、 $0 \sim 9 < A \sim Z < a \sim z$ 。

b、数字比字母要小。如“7” < “F”。

c、数字 0 要比数字 9 要小。并按 0 至 9 顺序递增。如“0” < “9”。

d、字母 A 比字母 Z 要小,并按 A 至 Z 顺序递增。如“A” < “Z”。

e、同个字母的大写字母比小写字母要小 32。如“A” < “a”。

需要记住几个常见字母的 ASCII 码大小:“A”为 65;“a”为 97;“0”为 48。

4.2.3 区别键代码和文字编码

在学习中,经常会把键代码和文字编码混淆,这是可以理解的,毕竟都是与文字息息相关的两种常见代码。但我们在使用时要时刻谨记它们是完全不同的。

在了解最常用的文字编码 ASCII 码后,继续了解一下键代码(也叫键盘扫描码)。

键盘上的每一个键都有两个唯一的数值进行标识。但为什么要用两个数值而不是一个数值呢?这是因为一个键可以被按下,也可以被释放。当一个键按下时,它们产生一个唯一的数值,当一个键被释放时,同样也会产生一个唯一的数值,我们把这些数值都保存在一张表里面,只要通过查表就可以知道哪一个键被单击,并且可以知道是被按下还是被释放了。而这些数值在系统中被称为键代码(键盘扫描码)。

通过以上可以看出,键代码是与键盘硬件相关的,而文字编码是与文字存储相关的。它们所在的领域和起到作用截然不同,如果将它们混淆,经常会得到和预想不同的结果。

4.2.4 文本操作命令

在编写程序时免不了对大量文本型的数据进行操作,文本操作类的命令比较全面,前面已经用到过“分割文本()”命令,下面介绍其他的常用文本操作命令。

1. “取文本长度()”命令

获取指定文本的字节长度,半角数字和字符为一个字节的长度,汉字和全角标点符号为两个字节的长度,如:

取文本长度(编辑框 1. 内容)

可以取出编辑框中文本的长度。

2. “取文本左边()”、“取文本右边()”和“取文本中间”命令

这3个命令可以取出一段文本中任意位置的文本。如:

取文本左边(编辑框 1. 内容,4)

可以将编辑框中的前4个字符取出来。

3. “寻找文本()”和“倒找文本()”命令

从当前文本中的指定位置开始寻找指定的文本,并返回最先找到该文本的位置。“寻找文本”是从指定文本的首部开始寻找,“倒找文本”相反。例如:

寻找文本(编辑框 1. 内容,“:”,1,假)

代码运行后会返回找到的第一个“:”的位置。

4. “文本替换()”命令

该命令可以将指定文本的某一部分用其他的文本替换。例如:

编辑框 1. 内容 = 文本替换(编辑框 1. 内容,4,2,“XX”)

将“编辑框 1”内容中第四个位置开始的2个字符替换成“XX”,并将结果显示在原编辑框中。

文本操作的命令通常都相互配合使用,参见例程4-1,其程序运行效果如图4-1所示。在按下按钮后,将编辑框中的IP地址,替换成“192.168.0.255”,并用信息框显示替换后的文本。

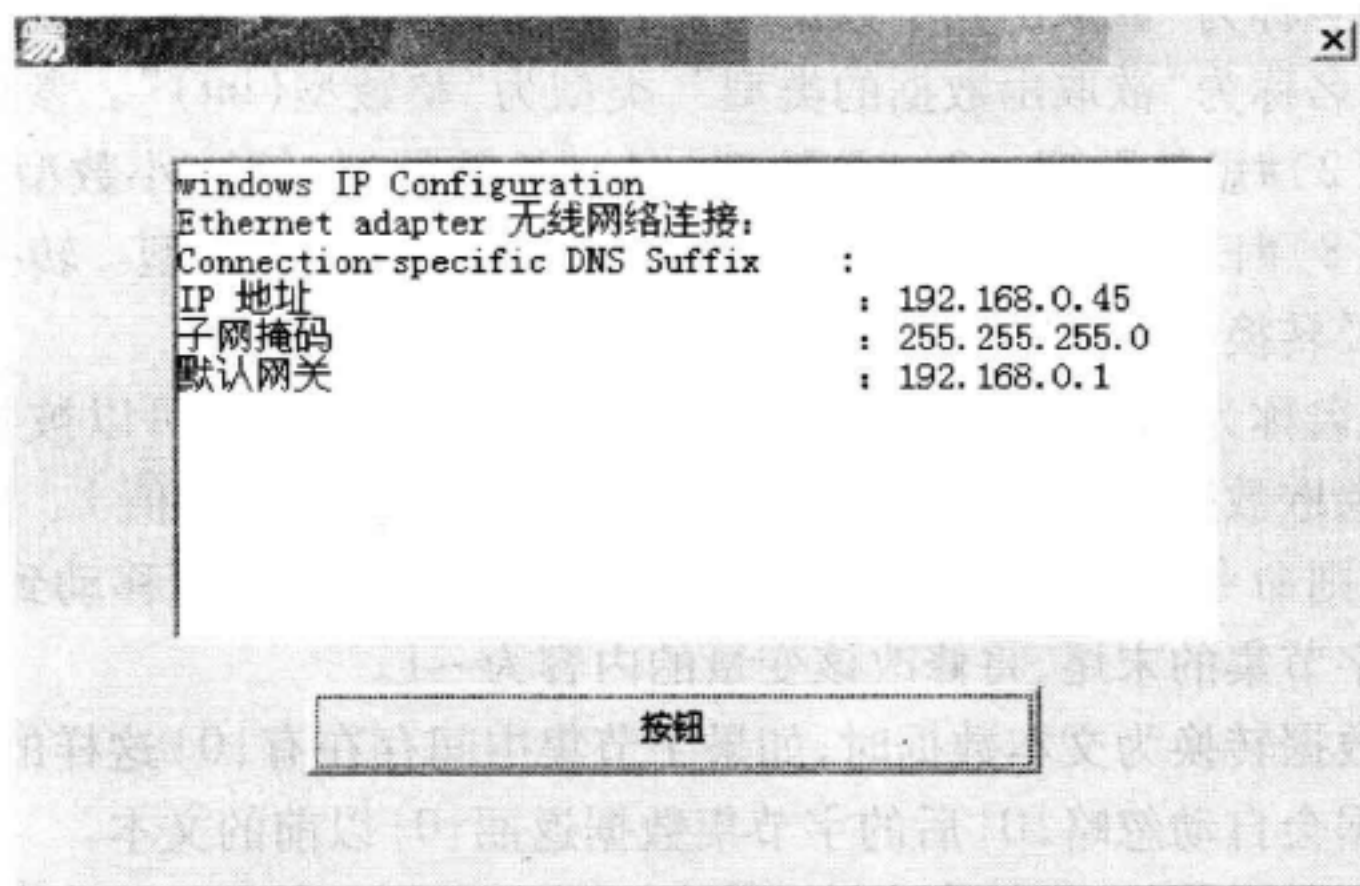


图4-1 编辑框中显示的文本

在替换前先要考虑将欲替换的文本的位置找到。IP地址是容易改变的,所以不能以数字串“192.168.0.45”作为查找对象。IP地址前有一个“:”,可以先找到这个“:”,然后加上“:”占用的2字符,就可以得到IP地址文本的第一个字符的位置,找到的这个位置,就可以确定被替换文本的位置。但IP地址前的“:”不是第一个“:”,所以要连续寻找2次第二次从“:”的下一个位置开始寻找,就可以找到正确的位置了。

要将替换后的文本取出来,只要得到替换后的文本长度,就可以得到。可以先查找“子网掩码”的“子”的位置,然后用“子”的位置减 IP 地址第一个字符的位置,就可以得到 IP 地址的文本长度。程序用到的代码如图 4-2 所示。

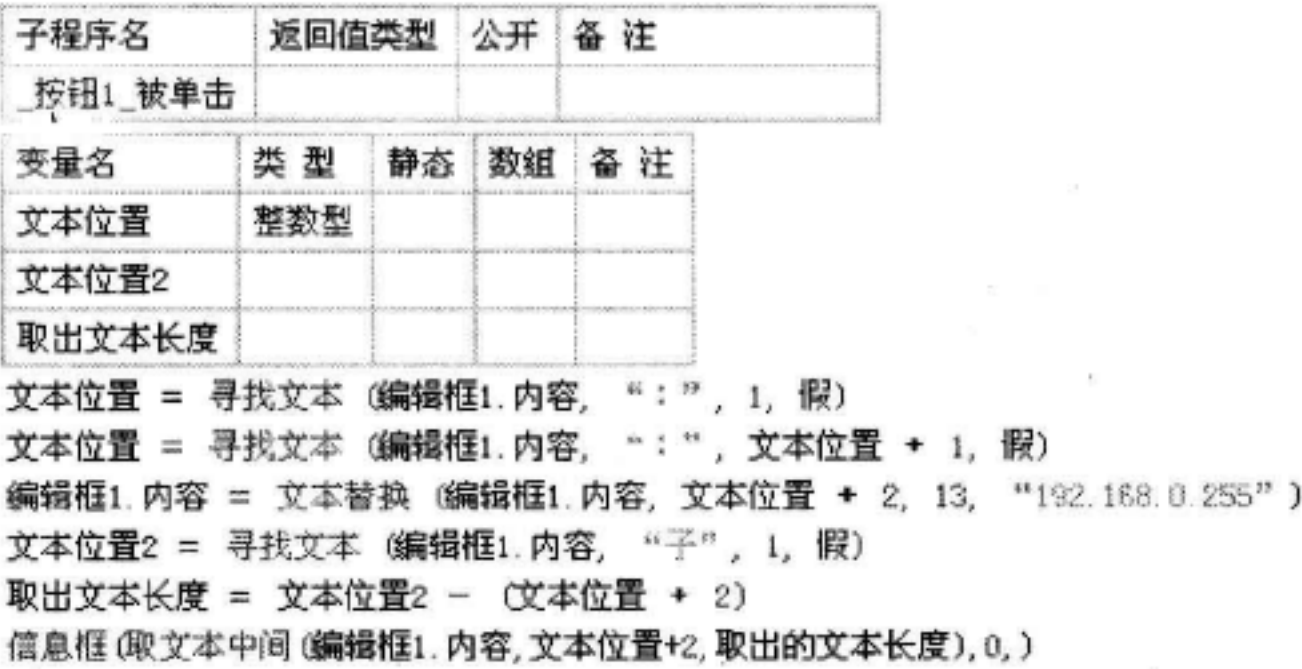


图 4-2 替换文本代码

4.2.5 字节集操作命令

说到文件的操作不得不说一下字节集这个数据类型,文件都是按照字节来表示的,而字节集其实就是字节型数组,易语言提供了多种字节集的操作,包括寻找替换分割等。我们这里简单介绍取字节集数据。

调用格式:〈通用型〉取字节集数据(字节集欲取出其中数据的字节集,整数型欲取出数据的类型,[整数型起始索引位置]) - 系统核心支持库 - > 字节集操作

英文名称: GetBinElement

取出字节集中指定位置指定数据类型的数据。本命令为初级命令。

参数<1>的名称为“欲取出其中数据的字节集”,类型为“字节集(bin)”。

参数<2>的名称为“欲取出数据的类型”,类型为“整数型(int)”。参数值可以为以下常量:(1)#字节型;(2)#短整数型;(3)#整数型;(4)#长整数型;(5)#小数型;(6)#双精度小数型;(7)#逻辑型;(8)#日期时间型;(9)#子程序指针型;(10)#文本型。转换后的数据将自动进行有效性校验及转换处理。

参数<3>的名称为“起始索引位置”,类型为“整数型(int)”,可以被省略。指定从字节集的什么地方开始取数据,索引值从 1 开始。如果被省略,默认为数值 1。如果为本参数提供一个整数型变量,则命令执行后将自动修改该变量内容,将其索引值移动到下一个读入位置。如果移动后到达字节集的末尾,将修改该变量的内容为 -1。

在将字节集数据转换为文本数据时,如果字节集中间存在有{0}这样的文本终止符,那么转换的字节集数据会自动忽略{0}后的字节集数据返回{0}以前的文本。

如果要填充一个字节集,最好先申请一个与要填充数据相等的空白字节集,然后按照数组赋值方式进行数据的写入,而不是简单的对字节集进行相加,这样会造成很大的进程延时。

4.3 磁盘操作

在 Windwos 编程时,还有一些与文件系统有关的功能也非常重要,包括添加、移动、删除文件或文件夹等。

4.3.1 目录路径

1. 绝对路径

在计算机中找寻需要的文件就必须知道文件的位置,而表示文件位置的方式就是路径。例如,路径:“E:\编程\易语言.exe”,可以知道“易语言.exe”文件是在E盘的“编程目录”中、类似这样完整描述文件位置的路径就是“绝对路径”。

2. 相对路径

“相对路径”就是由这个文件(或程序)所在的路径引起的跟其他文件(或文件夹)的路径关系。使用相对路径可以为我们带来非常多的便利,最大的好处就是在引用本程序相关的外部文件(如图片、声音)时,不再需要知道程序所在的完整路径就可以直接访问所需的外部文件。

3. 当前目录

当前目录是使用相对路径时的系统起始装载位置。当前目录程序中一个临时的变量,可以使用相关命令来获取和修改它的值。

4.3.2 磁盘操作命令

1. 磁盘相关命令

磁盘相关的主要函数有取磁盘总空间()、取磁盘剩余空间()、取磁盘卷标()、置磁盘卷标()、改变驱动器()等,具体见表4-1。

表 4-1 磁盘相关命令

函数名称	函数意义	说明
取磁盘总空间	返回以1024字节(1KB)为单位的指定磁盘空间	· 局部变量 磁盘空间,整数型 磁盘空间=取磁盘总空间(“C”)
取磁盘剩余空间	返回1024字节(1KB)为单位的指定磁盘现行剩余空间	· 局部变量 剩余空间,整数型 剩余空间=取磁盘剩余空间(“C”)
取磁盘卷标	返回指定磁盘的卷标文本	
置磁盘卷标	设置指定磁盘的卷标文本	
改变驱动器	改变当前的缺省驱动器	

2. 目录相关命令

目录相关的主要函数为改变目录()、取当前目录()、创建目录()、删除目录()等。具体见表4-2。

表 4-2 目录相关函数

函数名称	函数意义
改变目录	改变当前的目录,但不会改变缺省驱动器的位置
取当前目录	返回一个文本,用来代表当前的目录
创建目录	创建一个新的目录
删除目录	删除一个存在的目录及其中的所有子目录和下属文件

3. 文件相关命令

文件相关的主要函数为复制文件()、移动文件()、删除文件()、文件更名()、文件是否存在()、寻找文件()、取文件时间()、取文件尺寸()、取文件属性()、置文件属性()、取临时文件名()、读入文件()、写到文件()等,具体见表4-3。

表4-3 文件相关函数

函数名称	函 数 意 义
复制文件	对选中文件进行复制
移动文件	将文件从一个位置移动到另一个位置
删除文件	对选中文件进行删除
文件更名	重新命名一个文件和目录
文件是否存在	判断指定的磁盘文件是否真实存在。如存在返回真,否则返回假
寻找文件	返回一个文本,用以表示所找到的文件名或目录名,它必须与所要求的名称格式或文件属性相匹配。支持使用多字符(*)和单字符(?)通配符来指定多重文件
取文件时间	返回指定文件被创建或最后修改后的日期和时间。如果该文件不存在,将返回100年1月1日
取文件尺寸	返回一个文件的长度,单位是字节。如果该文件不存在,将返回-1
取文件属性	返回一个文件或目录的属性。此属性值由以下常量或其和构成:1. #只读文件;2. #隐藏文件;4. #系统文件;16. #子目录;32. #存档文件
置文件属性	为一个文件设置属性信息
取临时文件名	返回一个在指定目录中确定不存在的, TMP 全路径文件名称
读入文件	返回一个字节集,其中包含指定文件的所有数据
写到文件	本命令用作将一个或数个字节集顺序写到指定文件中,文件原有内容被覆盖

4.4 文件读写

文件就是保存在磁盘上的字节,采取什么样的结构去保存一个文件,即字节之间的关系。以及每个字节表示什么内容,是整数、文本或者其他数据,根据这些结构我们大体将文件分为两类。

(1) 顺序文件。普通的文本文件就是采用这样的顺序结构,顺序文件保存为一个连续的块,块中的字节代表的都是文本字符,读取和写入都是字符或字符类型的数据。

(2) 二进制文件。二进制文件中的字节代表任何东西,只有精确知道数据是如何写入到文件中后,才能对其进行正确的读取和检索,否则只能靠人工的猜测和推理。

顺序文件是最容易理解的,使用最广泛的文件类型,当需要处理典型文本编辑框所创建的文本文件时,应采用此方法。

顺序文件不太适合存储数字型的数据,因为每个数字都要按照文本方式存储,比如一个5位的数字字符,需要5个组的存储空间。而使用二进制方式,只需要2字节。

1. 打开文件命令

对文件做任何读写前都需要打开文件,打开文件使用打开文件语句。打开文件语法如下:

调用格式:〈整数型〉打开文件(文本型 欲打开的文件名称,[整数型 打开方式],[整数型 共享方式]) - 系统核心支持库 - > 文件读写

英文名称:open

打开一个普通文件,以对文件进行输入或输出。成功返回被打开文件的文件号,失败返回0。本命令为初级命令。

参数<1>的名称为“欲打开的文件名称”,类型为“文本型(text)”。

参数<2>的名称为“打开方式”,类型为“整数型(int)”,可以被省略。参数值说明对文件的操作方式,如果省略本参数,默认为“#读写”。方式值可以为以下常量之一:

- #读入:从指定文件读入数据,如果该文件不存在则失败;
- #写出:写出数据到指定文件,如果该文件不存在则失败;
- #读写:从文件中读入数据或者写出数据到文件,如果该文件不存在则失败;
- #重写:写出数据到指定文件。如果该文件不存在则先创建一个新文件,如果已经存在就先清除其中的所有数据;
- #改写:写出数据到指定文件。如果该文件不存在则创建一个新文件,如果已经存在就直接打开;
- #改读:从文件中读入数据或者写出数据到文件。如果该文件不存在则创建一个新文件,如果已经存在就直接打开。

参数<3>的名称为“共享方式”,类型为“整数型(int)”,可以被省略。参数值指定限制其他进程操作此文件的方式。如果省略本参数,默认为“#无限制”。方式值可以为以下常量之一:

- #无限制:允许其他进程任意读写此文件;
- #禁止读:禁止其他进程读此文件;
- #禁止写:禁止其他进程写此文件;
- #禁止读写:禁止其他进程读写此文件。

操作系统需求:Windows、Linux

2. 关闭文件命令

文件操作结束以后应该使用关闭文件语句来关闭文件。关闭文件语句可以关闭指定的文件,语法如下:

调用格式:〈无返回值〉关闭文件(整数型欲关闭的文件号) - 系统核心支持库 - > 文件读写

英文名称:close

关闭被打开的各种类型文件。本命令为初级命令。

参数<1>的名称为“欲关闭的文件号”,类型为“整数型(int)”。参数值指明欲关闭的文件号,该文件号由“打开文件”命令所返回。

操作系统需求:Windows、Linux

【范例4-1】制作一个简单的记事本软件。

下面简单地模拟一下WIN系统自带的记事本程序,实现记事本的打开文件、保存文件等基本操作,模拟记事本编辑框大小跟随窗口尺寸的改变而改变,代码在__启动窗口_尺寸被改变事件下,至于点关闭按钮弹出信息框等是在__启动窗口_可否被关闭事件下完成。对于控件发送消息暂不讨论,运行结果见图4-3,具体参看例程4-2。

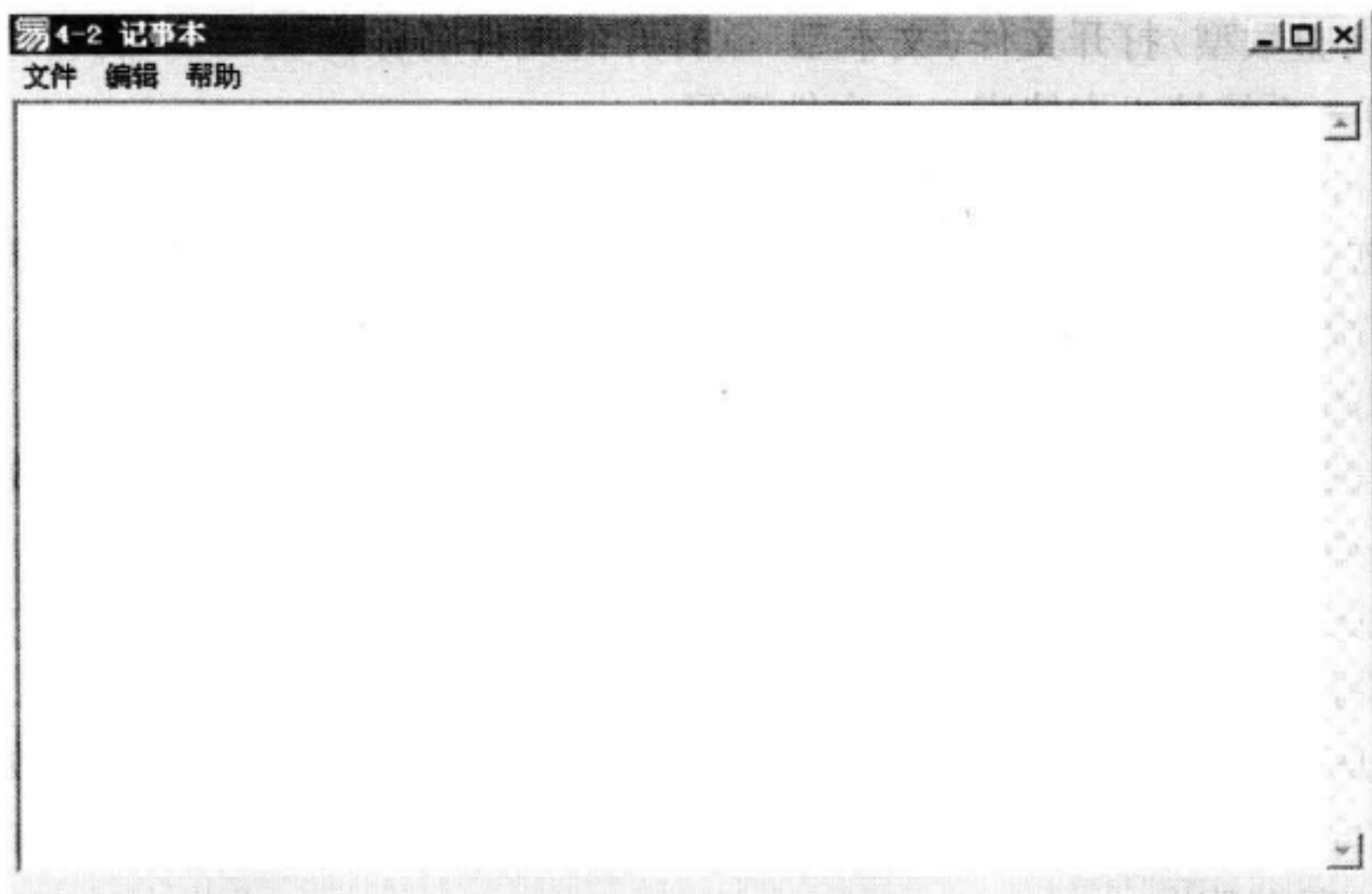


图 4-3 记事本软件

4.5 媒体播放

4.5.1 媒体播放格式

1. WAV

WAV 为微软公司(microsoft)开发的一种声音文件格式,它符合 RIFF(Rsource Interchange File Format)文件规范,用于保存 Windows 平台的音频信息资源,被 Windows 平台及其应用程序广泛支持,该格式也支持 msadpcm、CCITT A LAW 等多种压缩运算法,支持多种音频数字取样频率和声道,标准格式化的 WAV 文件和 CD 格式一样,也是 44.1kHz 的取样频率,16 位量化数字,因此在声音文件质量上和 CD 相差无几!

2. MIDI

MIDI(Musical Instrument Digital Interface)乐器数字接口,是 20 世纪 80 年代初为解决电声乐器之间的通信问题而提出的。MIDI 传输的不是声音信号,而是音符、控制参数等指令,它指示 MIDI 设备要做什么、怎么做,如演奏哪个音符,多大音量等。它们被统一表示成 MIDI 消息(MIDI message)。传输时采用异步串行通信,标准通信波特率为 $31.25 \times (1 \pm 0.01)$ KBaud。

3. MP3

MP3 全称为是动态影像专家压缩音频层面 3(Moving Picture Experts Group Audio Layer III),是当今比较流行的一种数字音频编码和有损压缩格式,它设计用来大幅度地降低音频数据量,而对于大多数用户来说重放的音质和最初的不压缩音频相比没有明显的下降。它是在 1991 年由于位于德国埃尔朗根的研究组织 fraunhofer - gesellschaft 的一组工程师发明和标准化的。

4.5.2 媒体播放命令

1. “播放音乐()”

可以播放 .WAV、.MID 声音文件或相应格式的字节集声音数据、声音资源。

实例代码如下：

逻辑变量 = 播放音乐("C:\Windows\音乐文件.WAV",真)

或逻辑变量 = 播放音乐(#声音文件,真)

其中“#声音文件”为声音资源。

第一个参数值为.WAV、.MID 声音文件或相应格式字节集声音数据、声音资源。

第二个参数值为“真”表示指定音乐将被循环播放,否则仅播放一次。如果本参数被省略,默认为仅播放一次。

2. “播放 MID()”命令

可以自动连续播放多个 MIDI 声音文件(注意不支持 WAV)或相应格式字节集声音数据、声音资源。

实例代码如下：

播放 MID(, , 读入文件("C:\Windows\音乐文件.MID"),#声音文件)

或加入成员(字节集数组变量,读文件("C:\Windows\音乐文件.MID"))

加入成员(字节集数组变量,#声音文件)

播放 MID(, , 读入文件("C:\Windows\音乐文件.MID"),字节集数组变量)

其中,第一个参数为播放次数,第二个参数为“间隔时间”,第三个参数为“欲播放的 MIDI 音乐”。被播放的音乐文件可以连续扩充,并按顺序播放。这里的音乐文件必须转换为字节集型数据,也可以提供保存了多个字节集型音乐文件的字节集数组。

4.6 系统处理

4.6.1 了解剪切板

剪切板是指 Windows 操作系统提供的一个暂存数据,并且提供共享的一个模块。也称为数据中转站,剪切板在后台起作用,在内存里,是操作系统设置的一段存储区域,在硬盘中是找不到的。只要在文本输入的地方按 CTRL + V 或右键“粘贴”剪切板中的内容就出现了。新的内容送到剪切板后,将覆盖旧内容。即剪切板只保存当前的一份内容在内存里,所以电脑关闭或重启,存在剪切板中的内容将丢失。

4.6.2 了解注册表

注册表(Registry)是 Windows 中的一个重要的数据库,用于存储系统和应用程序的设置信息。注册表是 Windows 程序员建造的一个复杂的信息数据库。它是多层次式的。不同 Windows 版本注册表的基本结构相同。其中的复杂数据会以不同方式上结合,从而产生出一个绝对唯一的注册表。

计算机配置和缺省用户设置的注册表数据在 winnt 核心系统中被保存在 default,sam,security,software、system、ntuser.dat。

注册表由键(或“项”)、子键(子项)和值项构成。一个键就是分支中的一个文件夹,而子

键就是这个文件夹中的子文件夹。子键同样是一个键,一个值项则是一个键的当前定义,由名称、数据类型以及分配的值组成。一个键可以有一个或多个值,每个值的名称各不相同,如果一个值的名称为空,则该值为该键的默认值。

4.6.3 系统处理命令

1. 运行

调用格式:〈逻辑型〉运行(欲运行的命令行,是否等待程序运行完毕,[被运行程序窗口显示方式])

英文名称:run

本命令运行指定的可执行 EXE 文件或者外部命令。如果成功,返回真,否则返回假。本命令为初级命令。

参数<1>的名称为“欲运行的命令行”,类型为“文本型(text)”。

参数<2>的名称为“是否等待程序运行完毕”,类型为“逻辑型(bool)”,初始值为“假”。

参数<3>的名称为“被运行程序窗口显示方式”,类型为“整数型(int)”,可以被省略。

参数值可以为以下常量之一:(1)#隐藏窗口;(2)#普通激活;(3)#最小化激活;(4)#最大化激活;(5)#普通不激活;(6)#最小化不激活。如果省略本参数,默认为“普通激活”方式。

※取剪辑板文本

调用格式:〈文本型〉取剪辑板文本()

英文名称:GetClipboardText

返回存放于当前 Windows 系统剪辑板中的文本。如果当前剪辑板中无文本数据,将返回空文本。本命令为初级命令。

2. 置剪辑板文本

调用格式:〈逻辑型〉置剪辑板文本(准备置入剪辑板的文本)

英文名称:SetClipboardText

将指定文本存放到当前 Windows 系统剪辑板中去,剪辑板中的原有内容被覆盖。成功返回真,失败返回假。本命令为初级命令。

参数<1>的名称为“准备置入剪辑板的文本”,类型为“文本型(text)”。

3. 剪辑板中可有文本

调用格式:〈逻辑型〉剪辑板中可有文本()

英文名称:IsHaveTextInClip

如果当前 Windows 系统剪辑板中有文本数据,则返回真,否则返回假。本命令为初级命令。

4. 清除剪辑板

调用格式:〈无返回值〉清除剪辑板()

英文名称:ClearClipboard

清除当前 Windows 系统剪辑板中的所有数据。本命令为初级命令。

5. 取屏幕宽度

调用格式:〈整数型〉取屏幕宽度()

英文名称:GetScreenWidth

返回屏幕当前显示区域的宽度,单位为像素点。本命令为初级命令。

6. 取屏幕高度

调用格式:〈整数型〉取屏幕高度()

英文名称: GetScreenHeight

返回屏幕当前显示区域的高度,单位为像素点。本命令为初级命令。

7. 取鼠标水平位置

调用格式:〈整数型〉取鼠标水平位置()

英文名称: GetCursorHorzPos

返回鼠标指针的当前水平位置,单位为像素点,相对于屏幕左边。本命令为初级命令。

8. 取鼠标垂直位置

调用格式:〈整数型〉取鼠标垂直位置()

英文名称: GetCursorVertPos

返回鼠标指针的当前垂直位置,单位为像素点,相对于屏幕顶边。本命令为初级命令。

9. 取颜色数

调用格式:〈整数型〉取颜色数()

英文名称: GetColorCount

返回当前显示方式所提供的最大颜色显示数目。本命令为初级命令。

10. 输入框

调用格式:〈逻辑型〉输入框([提示信息],[窗口标题],[初始文本],存放输入内容的容器,[输入方式])

英文名称: InputBox

在一对话框中显示提示,等待用户输入正文并按下按钮。如果用户在确认输入后(按下“确认输入”按钮或回车键)退出,返回真,否则返回假。本命令为初级命令。

参数<1>的名称为“提示信息”,类型为“文本型(text)”,可以被省略。如果提示信息包含多行,可在各行之间用回车符(即“字符(13)”)、换行符(即“字符(10)”)或回车换行符的组合(即“字符(13)+字符(10)”)来分隔。如果提示信息太长或行数过多,超过部分将不会被显示出来。

参数<2>的名称为“窗口标题”,类型为“文本型(text)”,可以被省略。参数值指定显示在对话框标题栏中的文本。如果省略,默认为文本“请输入:”。

参数<3>的名称为“初始文本”,类型为“文本型(text)”,可以被省略。参数值指定初始设置到对话框输入文本框中的内容。

参数<4>的名称为“存放输入内容的容器”,类型为“通用型(all)”,提供参数数据时只能提供容器。参数值所指定的容器可以为数值或文本型,用于以不同的数据类型取回输入内容。

参数<5>的名称为“输入方式”,类型为“整数型(int)”,可以被省略。参数值可以为以下常量值:(1)#输入文本;(2)#输入整数;(3)#输入小数;(4)#输入密码。如果省略本参数,默认为“#输入文本”。

11. 信息框

调用格式:〈整数型〉信息框(提示信息,按钮,[窗口标题])

英文名称:MsgBox

在对话框中显示信息,等待用户单击按钮,并返回一个整数告诉用户单击哪一个按钮。该整数为以下常量值之一:(0)#确认钮;(1)#取消钮;(2)#放弃钮;(3)#重试钮;(4)#忽略钮;(5)#是钮;(6)#否钮。如果对话框有“取消”按钮,则按下 ESC 键与单击“取消”按钮的效果相同。本命令为初级命令。

参数<1>的名称为“提示信息”,类型为“通用型(all)”。提示信息只能为文本、数值、逻辑值或日期时间。如果提示信息为文本且包含多行,可在各行之间用回车符(即“字符(13)”)、换行符(即“字符(10)”)或回车换行符的组合(即“字符(13)+字符(10)”)来分隔。

参数<2>的名称为“按钮”,类型为“整数型(int)”,初始值为“0”。参数值由以下几组常量值组成,在将这些常量值相加以生成参数值时,每组值只能取用一个数字(第五组除外):

第一组(描述对话框中显示按钮的类型与数目):

(0)#确认钮;(1)#确认取消钮;(2)#放弃重试忽略钮;(3)#取消是否钮;(4)#是否钮;(5)#重试取消钮。

第二组(描述图标样式):

(16)#错误图标;(32)#询问图标;(48)#警告图标;(64)#信息图标

第三组(说明哪一个按钮是缺省默认值):

(0)#默认按钮一;(256)#默认按钮二;(512)#默认按钮三;(768)#默认按钮四

第四组(决定如何等待消息框结束):

(0)#程序等待;(4096)#系统等待

第五组(其他):

(65536)#位于前台;(524288)#文本右对齐

参数<3>的名称为“窗口标题”,类型为“文本型(text)”,可以被省略。参数值指定显示在对话框标题栏中的文本。如果省略,默认为文本“信息:”。

12. 鸣叫

调用格式:〈无返回值〉鸣叫()

英文名称:beep

通过计算机媒体设备或者喇叭发出一个声音。本命令为初级命令。

13. 取启动时间

调用格式:〈整数型〉取启动时间()

英文名称:GetTickCount

返回自 Windows 系统启动后到现在为止所经历过的毫秒数。本命令为初级命令。

14. 置等待鼠标

调用格式:〈无返回值〉置等待鼠标()

英文名称:SetWaitCursor

本命令设置现行鼠标的形状为沙漏形,用作在即将长时间执行程序前提示操作者。本命令为初级命令。

15. 恢复鼠标

调用格式:〈无返回值〉恢复鼠标()

英文名称:RestroeCursor

本命令恢复现行鼠标的原有形状,用作与“置等待鼠标”命令配对使用。本命令为初级命令。

16. 延时

调用格式:〈无返回值〉延时(欲等待的时间)

英文名称:sleep

本命令暂停当前程序的运行并等待指定的时间。本命令为初级命令。

参数<1>的名称为“欲等待的时间”,类型为“整数型(int)”。本参数指定欲暂停程序执行的时间,单位为毫秒。

17. 取文本注册项

调用格式:〈文本型〉取文本注册项(根目录,全路径注册项名,[默认文本])

英文名称:GetTextRegItem

在 Windows 注册表中返回指定的文本类型注册表项值。如欲读取注册项默认值,请在项目名后加“\”号,如“test\”。与“取文本注册表项”命令不同的是本命令可以取任意位置处的注册表项。本命令为中级命令。

参数<1>的名称为“根目录”,类型为“整数型(int)”。可以为以下常量值之一:(1)#根类;(2)#现行设置;(3)#现行用户;(4)#本地机器;(5)#所有用户。

参数<2>的名称为“全路径注册项名”,类型为“文本型(text)”。

参数<3>的名称为“默认文本”,类型为“文本型(text)”,可以被省略。如果指定的注册表项不存在,将返回此默认文本。如果指定的注册表项不存在且本参数被省略,将返回一个长度为0的空文本。

18. 取数值注册项

调用格式:〈整数型〉取数值注册项(根目录,全路径注册项名,[默认数值])

英文名称:GetNumRegItem

在 Windows 注册表中返回指定的数值类型注册表项值。如欲读取注册项默认值,请在项目名后加“\”号,如“test\”。与“取数值注册表项”命令不同的是本命令可以取任意位置处的注册表项。本命令为中级命令。

参数<1>的名称为“根目录”,类型为“整数型(int)”。可以为以下常量值之一:(1)#根类;(2)#现行设置;(3)#现行用户;(4)#本地机器;(5)#所有用户。

参数<2>的名称为“全路径注册项名”,类型为“文本型(text)”。

参数<3>的名称为“默认数值”,类型为“整数型(int)”,可以被省略。如果指定的注册表项不存在,将返回此默认数值。如果指定的注册表项不存在且本参数被省略,将返回数值0。

19. 取字节集注册项

调用格式:〈字节集〉取字节集注册项(根目录,全路径注册项名,[默认字节集])

英文名称:GetBinRegItem

在 Windows 注册表中返回指定的字节集类型注册表项值。如欲读取注册项默认值,请在项目名后加“\”号,如“test\”。与“取字节集注册表项”命令不同的是本命令可以取任意位置处的注册表项。本命令为中级命令。

参数<1>的名称为“根目录”,类型为“整数型(int)”。可以为以下常量值之一:(1)#根

类;(2)#现行设置;(3)#现行用户;(4)#本地机器;(5)#所有用户。

参数<2>的名称为“全路径注册项名”,类型为“文本型(text)”。

参数<3>的名称为“默认字节集”,类型为“字节集(bin)”,可以被省略。如果指定的注册表项不存在,将返回此默认字节集。如果指定的注册表项不存在且本参数被省略,将返回空字节集。

20. 写注册项

调用格式:〈逻辑型〉写注册项(根目录,全路径注册项名,欲写入值)

英文名称:SaveRegItem

在 Windows 注册表中保存或建立指定的注册表项。如欲写入注册项默认值,请在项目名后加“\”号,如“test\”。成功返回真,否则返回假。与“写注册表项”命令不同的是本命令可以写任意位置处的注册表项。本命令为中级命令。

参数<1>的名称为“根目录”,类型为“整数型(int)”。可以为以下常量值之一:(1)#根类;(2)#现行设置;(3)#现行用户;(4)#本地机器;(5)#所有用户。

参数<2>的名称为“全路径注册项名”,类型为“文本型(text)”。

参数<3>的名称为“欲写入值”,类型为“通用型(all)”。参数值指定欲写入到指定注册表项中的值,只能为数值、文本或者字节集,否则命令将失败。

21. 删除注册项

调用格式:〈逻辑型〉删除注册项(根目录,全路径注册项名)

英文名称>DeleteRegItem

在 Windows 注册表中删除指定注册表项或注册表目录。如欲删除注册项默认值,请在项目名后加“\”号,如“test\”。成功返回真,否则返回假。与“删除注册表项”命令不同的是本命令可以删除任意位置处的注册表项或目录。注意在删除目录之前必须先删除该目录下所有的项目。本命令为中级命令。

参数<1>的名称为“根目录”,类型为“整数型(int)”。可以为以下常量值之一:(1)#根类;(2)#现行设置;(3)#现行用户;(4)#本地机器;(5)#所有用户。

参数<2>的名称为“全路径注册项名”,类型为“文本型(text)”。

22. 注册项是否存在

调用格式:〈逻辑型〉注册项是否存在(根目录,全路径注册项名)

英文名称:IsRegItemExist

如果指定注册表项存在,返回真,否则返回假。如欲检查注册项是否有默认值,请在项目名后加“\”号,如“test\”。本命令为中级命令。

参数<1>的名称为“根目录”,类型为“整数型(int)”。可以为以下常量值之一:(1)#根类;(2)#现行设置;(3)#现行用户;(4)#本地机器;(5)#所有用户。

参数<2>的名称为“全路径注册项名”,类型为“文本型(text)”。

23. 取默认底色

调用格式:〈整数型〉取默认底色()

英文名称:GetBackColor

取回 Windows 系统的默认窗口背景颜色。本命令为初级命令。

24. 快照

调用格式:〈字节集〉快照([窗口句柄],[输出宽度],[输出高度])

英文名称: GetWinPic

捕获指定窗口或屏幕上所有现有显示内容,返回相应图片数据。如果失败,返回空字节集。本命令为高级命令。

参数<1>的名称为“窗口句柄”,类型为“整数型(int)”,可以被省略。指定欲捕获其显示内容的窗口。如果被省略,默认为捕获屏幕显示内容。

参数<2>的名称为“输出宽度”,类型为“整数型(int)”,可以被省略。指定图片的输出宽度。如果小于0,参数值指定的是最终图片输出宽度相对于所取得图片宽度的百分比(最小为10%);如果等于0,则按图片原宽度输出;如果大于0,指定输出图片的绝对宽度。如果本参数被省略,默认值为0。

参数<3>的名称为“输出高度”,类型为“整数型(int)”,可以被省略。指定图片的输出高度。如果小于0,参数值指定的是最终图片输出高度相对于所取得图片高度的百分比(最小为10%);如果等于0,则按图片原高度输出;如果大于0,指定输出图片的绝对高度。如果本参数被省略,默认值为0。

25. 读配置项

调用格式:〈文本型〉读配置项(配置文件名,节名称,配置项名称,[默认文本])

英文名称: GetKeyText

读取指定配置文件中指定项目的文本内容。本命令为初级命令。

参数<1>的名称为“配置文件名”,类型为“文本型(text)”。指定配置文件的名称,通常以.ini作为文件名后缀。

参数<2>的名称为“节名称”,类型为“文本型(text)”。包含欲读入配置项所处节的名称。

参数<3>的名称为“配置项名称”,类型为“文本型(text)”。参数值指定欲读入配置项在其节中的名称。

参数<4>的名称为“默认文本”,类型为“文本型(text)”,可以被省略。如果指定配置项不存在,将返回此默认文本。如果指定配置项不存在且本参数被省略,将返回空文本。

26. 写配置项

调用格式:〈逻辑型〉写配置项(配置文件名,节名称,[配置项名称],[欲写入值])

英文名称: SetKeyText

将指定文本内容写入指定配置项中或者删除指定的配置项或节,如果指定配置文件不存在,将会自动创建。成功返回真,失败返回假。本命令为初级命令。

参数<1>的名称为“配置文件名”,类型为“文本型(text)”。指定配置文件的名称,通常以.ini作为文件名后缀。

参数<2>的名称为“节名称”,类型为“文本型(text)”。包含欲写入配置项所处节的名称。

参数<3>的名称为“配置项名称”,类型为“文本型(text)”,可以被省略。参数值指定欲写入配置项在其节中的名称。如果参数值被省略,则删除指定节及其下的所有配置项。

参数<4>的名称为“欲写入值”,类型为“文本型(text)”,可以被省略。参数值指定欲写

入到指定配置项中的文本。如果参数值被省略,则删除所指定配置项。

27. 取配置节名

调用格式:〈文本型数组〉取配置节名(配置文件名)

英文名称: GetSectionNames

返回指定配置文件中所有已有节名的文本数组。本命令为初级命令。

参数<1>的名称为“配置文件名”,类型为“文本型(text)”。指定配置文件的名称,通常以.ini作为文件名后缀。

28. 取操作系统类别

调用格式:〈整数型〉取操作系统类别()

英文名称: GetSysVer

返回当前 Windows 系统的版本类别。返回值为以下值之一:0、未知;1、Windows32S;2、Windows9X(包含 Win95、Win98、WinME 等);3、WindowsNT(包含 WinNT、Win2000、WinXP 等)。本命令为初级命令。

第5章 主体组件

本章首先介绍未能在组件面板中列出的几种组件:窗口组件、菜单组件、信息框组件、输入框组件、数据库维护者等。类似于这样的组件,随着一些支持库的安装,会不断增加。

其中,窗口组件是基本的组件,它拥有易语言中最基本的属性事件与组件命令。

5.1 窗口组件

为什么窗口也算是一个组件呢?在其他的语言类书籍中,有的将窗口命名为窗体,一切组件都是放在窗体中。而在本书中,因为窗口有着与其他组件一样的属性、事件等,所以从这个意义上它等同于一个组件。在易语言中,窗口是最基本的组件,其他组件必需放在窗口中使用,因此首先要介绍的就是窗口组件。

窗口组件是程序界面可视化操作的基本单元,也是程序中主要的输入/输出部件。所有的组件都是一个独立的对象,拥有自己的属性、事件和方法。易语言可视化界面设计与良好的对象封装保证了易程序开发的便捷与高效。

易语言在默认设置情况下,每当新建一个易程序窗口程序,都会自动对应生成一个“_启动窗口”组件。它有自己的属性、事件和方法,并且可以作为其他控件的载体和容器。在易语言中不能从组件箱中直接新增窗体。新建窗口可使用菜单或在程序面板中使用鼠标右键添加。

5.1.1 窗口概述

双击易语言图标,运行它。首先,会启动一个名为“新建”的对话框,如图5-1所示。

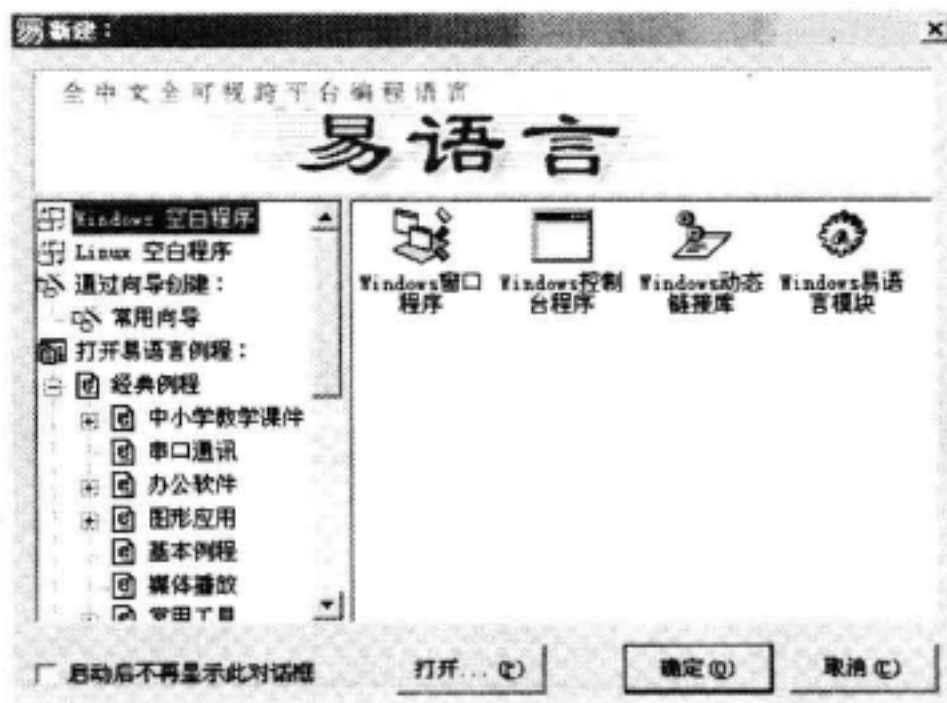


图5-1 “新建”对话框

如果将“新建”对话框左下角的“启动后不再显示此对话框”复选框去除,在每次启动易语言时,将不再弹出本对话框。

在此,请选中“新建”选项卡中左上角的“Windows 窗口程序”图标,单击“确定”按钮后,会

自动生成一个名为“_启动窗口”的空白窗口,如图 5-2 所示。左边的程序面板中已经有一个“_启动窗口”窗口列表,中间即是一个空白的窗口。

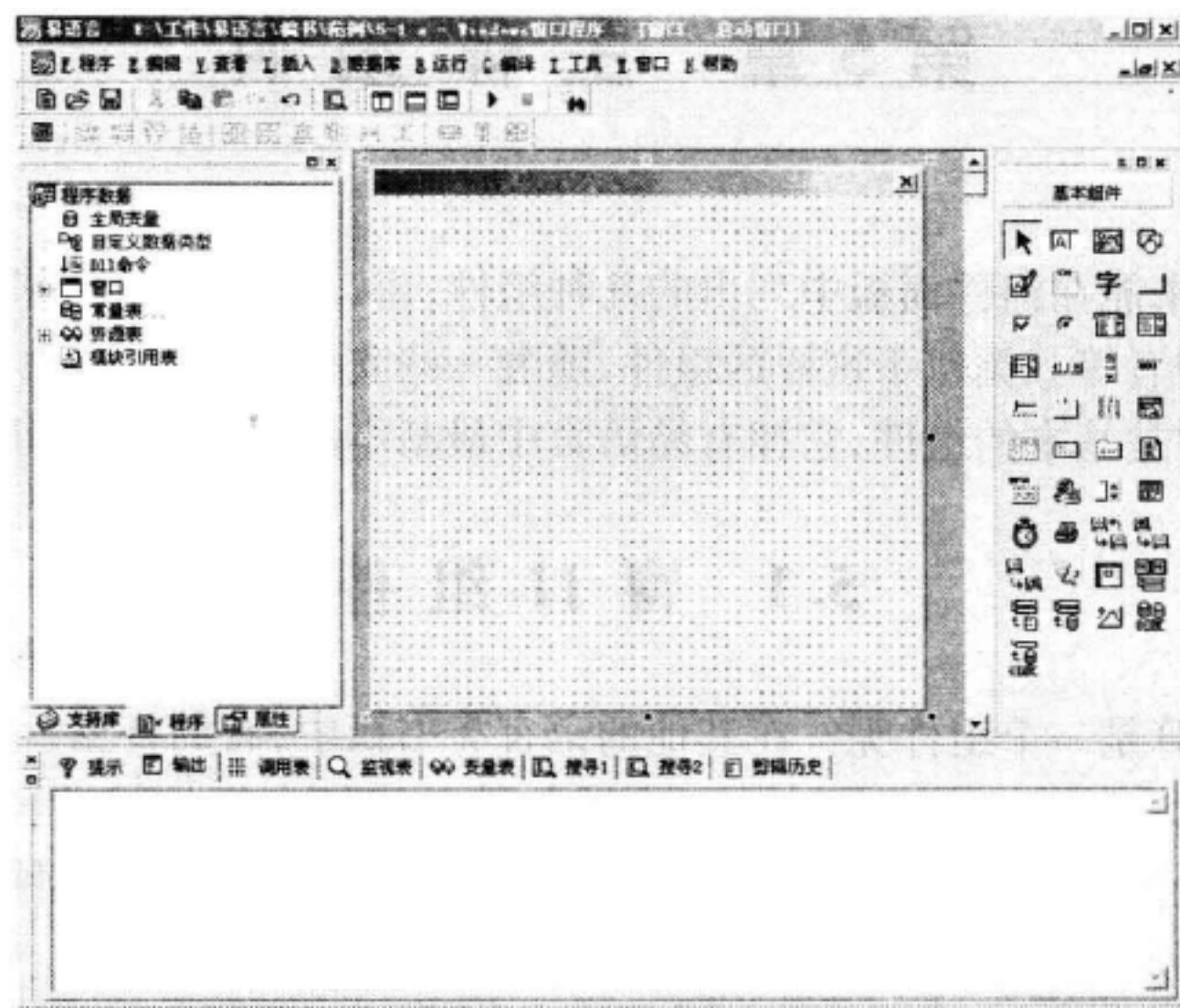



图 5-2 新建后生成的“_启动窗口”

现在可以直接运行它。点击易语言系统主菜单“运行”→“运行”或使用热键 F5,或使用工具栏上的运行按钮。运行的结果只有一个“_启动窗口”,如图 5-3 所示。

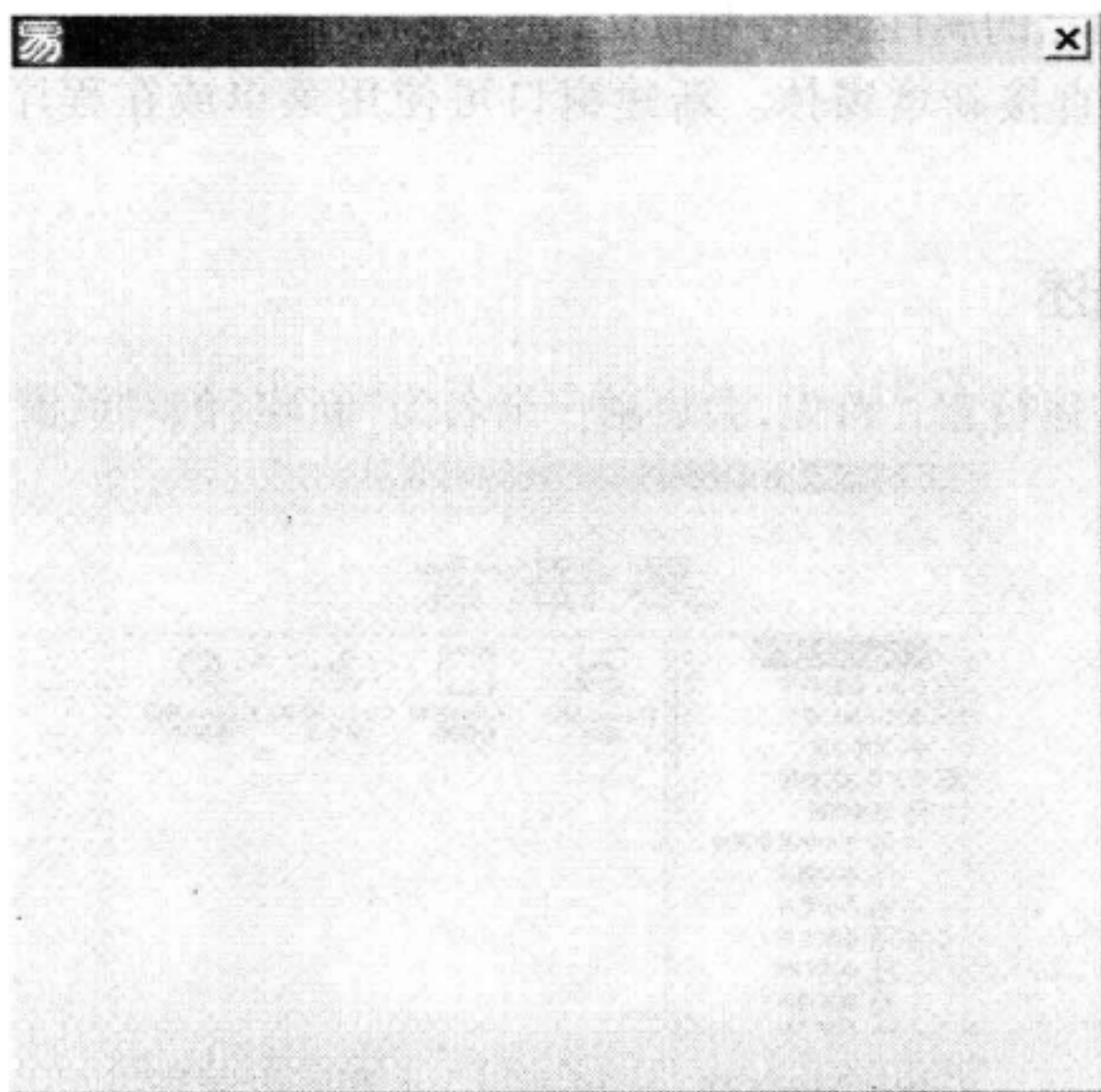






图 5-3 运行程序时的效果

可以观察到,这个窗口内几乎什么都没有,既没有标题,也没有熟悉的最大化按钮与最小化按钮,窗口内也没有任何文字和图片。但是,它是可以运行的,运行的结果就是弹出一个空白的窗口。

因此,一个空白的窗口就是一个程序,它虽然什么也没有做,但是它已经满足了程序的第

一要素:有一个载体,那就是窗口,以后可以在这个窗口中添加文字、图片、按钮等,还可以根据鼠标所单击的按钮等,进行一系列的判断,以及显示判断后的结果。

如果要结束这个空白窗口程序,可以有以下几种方法:

- 点击窗口中的  按钮。
- 按键盘上的 Esc 键。
- 按键盘上的 Alt + F4 组合键。
- 用鼠标单击易语言编程系统中的终止  按钮。

下面,对该程序进行简单设置。单击左侧的属性面板,找到“标题”属性,在里面填写“易语言练习”5个字,并且回车或随便单击其他位置。这时可以看到原空白窗口的标题栏上已出现“易语言练习”这5个字,如图5-4所示。

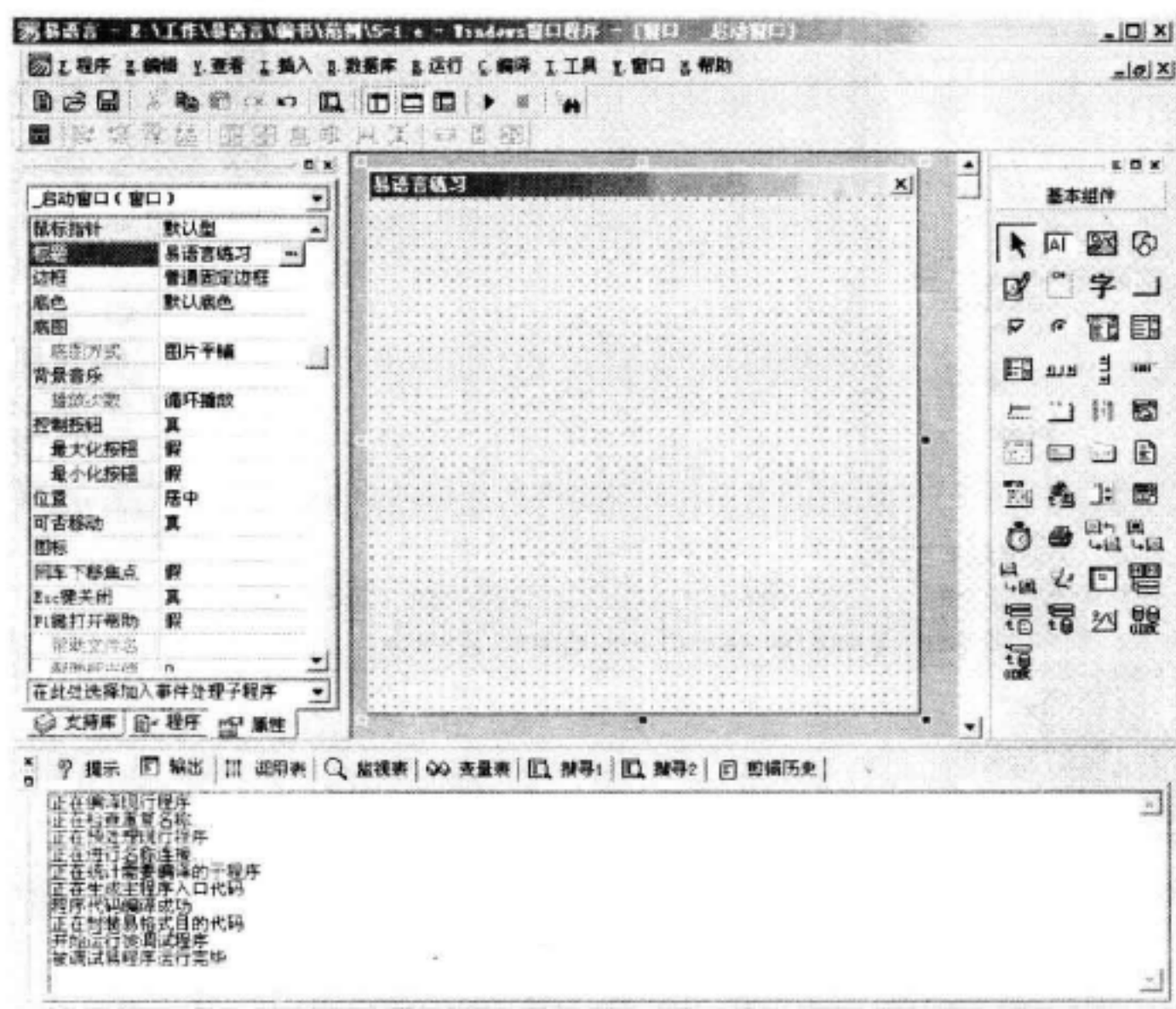


图 5-4 为空白窗口增加标题

左侧的属性面板中,刚刚改变的“标题”即为窗口的属性之一,只要改变属性,就可以控制窗口的状态,如窗口的位置、尺寸,就是由“左边”、“顶边”、“宽度”、“高度”属性决定的,其他还有“可视”属性控制窗口是否可见,“禁止”属性控制窗口是否可操作,“边框”属性可以控制窗口边框的形态,“底图”属性可以为窗口配上不同的图片……总之,可以尝试改变属性面板中的属性内容,再试运行一下,以体会这些属性设置后对窗口的影响。

5.1.2 窗口的属性

窗口是特殊的组件。说它特殊是因为它没有列在组件面板中。插入新窗口要选择易语言系统主菜单“插入”→“窗口”才可以新建其他窗口。但它却和普通组件一样,拥有自己的属性、事件和组件命令。而且您将会发现,组件的命令,全都是“窗口”组件的命令。或者反过来说,编辑框、按钮、标签等这些可视组件,都是广义上的“窗口”。每一个程序,至少有一个窗口,可见掌握它是有必要的。



图 5-6 设置控制按钮、最大化按钮、最小化按钮的属性为真



图 5-7 改变窗口边框形态

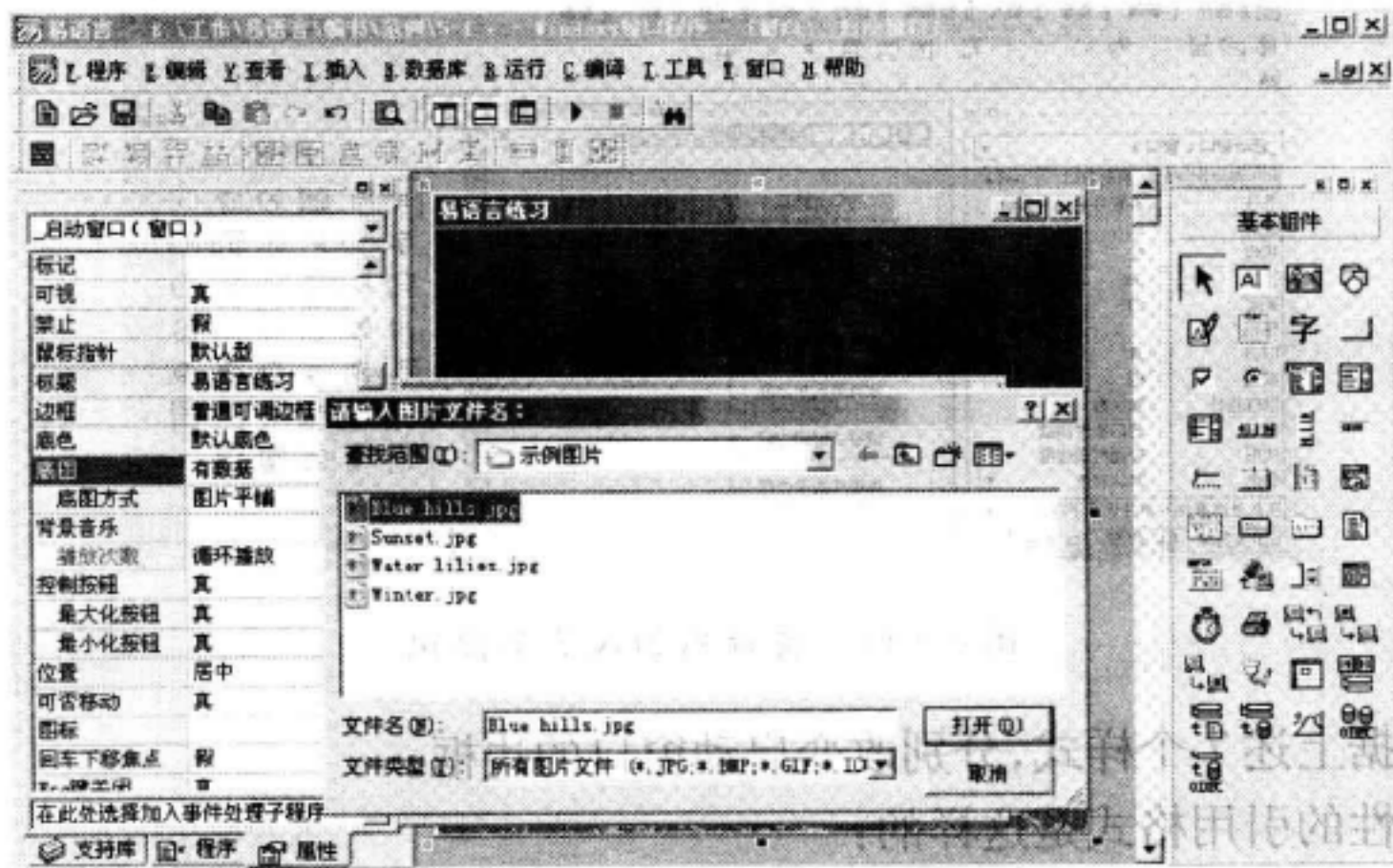


图 5-8 为窗口加入底图

“底色”属性虽然激活时也有图标,但它是一个常量型属性,不是字节集属性,字节集属性在有数据时会有“有数据”三个字的提示,这是它的唯一标志。

如果不想在窗口中显示底图,可将底图删除。删除“底图”属性的内容,只需要在属性上单击鼠标右键,在弹出小菜单中选其中的“删除内容”,或者在激活后直接用[Del]键删除。如果用弹出菜单中的“写到文件”项,还可以将内容导出为原始的文件,如图5-9所示。

单击“底色”属性后会弹出调色板对话框,从中选择自己喜欢的颜色作为窗口的底色,如图5-10所示。

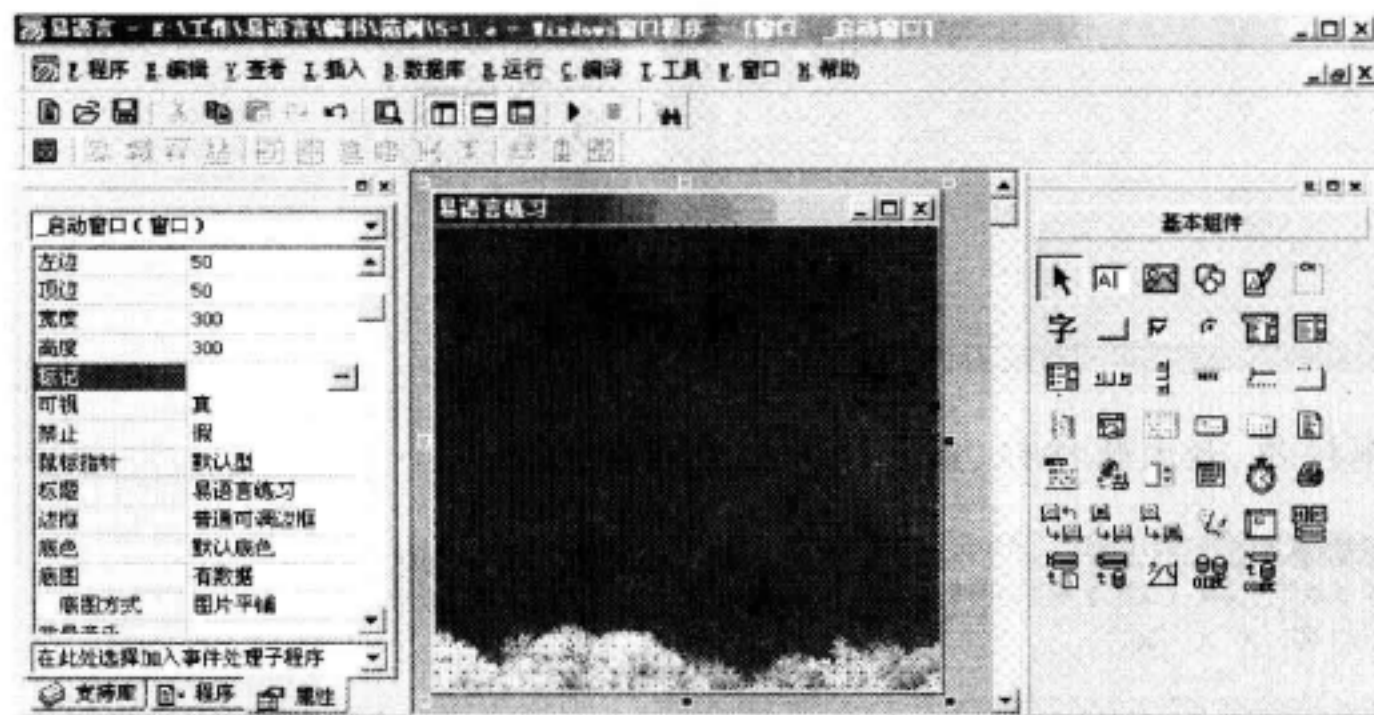


图 5-9 删除底图内容




图 5-10 设置“底色”属性的调色板

以上内容是在窗口编辑前预先设置好窗口的大小,底图、按钮、底色等内容,如何在程序中也可以用命令来实现上述的功能? 请继续看下面的介绍。

2. 用程序控制窗口属性

窗口的边框有以下7种可选样式:“0. 无边框”、“1. 普通可调边框”、“2. 普通固定边框”、“3. 窄标题可调边框”、“4. 窄标题固定边框”、“5. 镜框式可调边框”、“6. 镜框式固定边框”等。默认值为“2. 普通固定边框”。

在组件面板中找到按钮组件,在启动窗口内拉出7个按钮,并且将按钮的标题改为上述7个样式的相关名字,如图5-11所示。

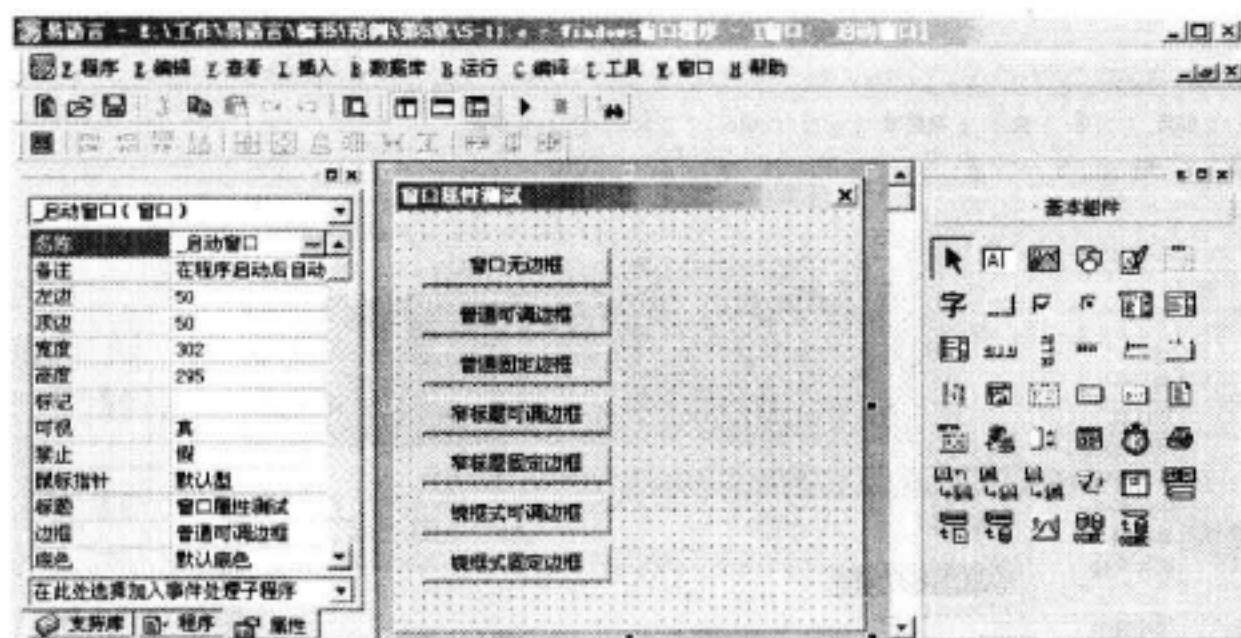


图 5-11 窗口内加入7个按钮

下面就要根据上述7个样式,分别改变启动窗口的边框。

启动窗口属性的引用格式是这样的:

启动窗口. 边框 = X

其中的X可以用上述0-6的数值代替。

双击第1个按钮,这时会进入程序的设计界面。会产生一个“_按钮1_被单击”的子程序,在下面输入一行命令,如图5-12所示。

_启动窗口.边框 = 0

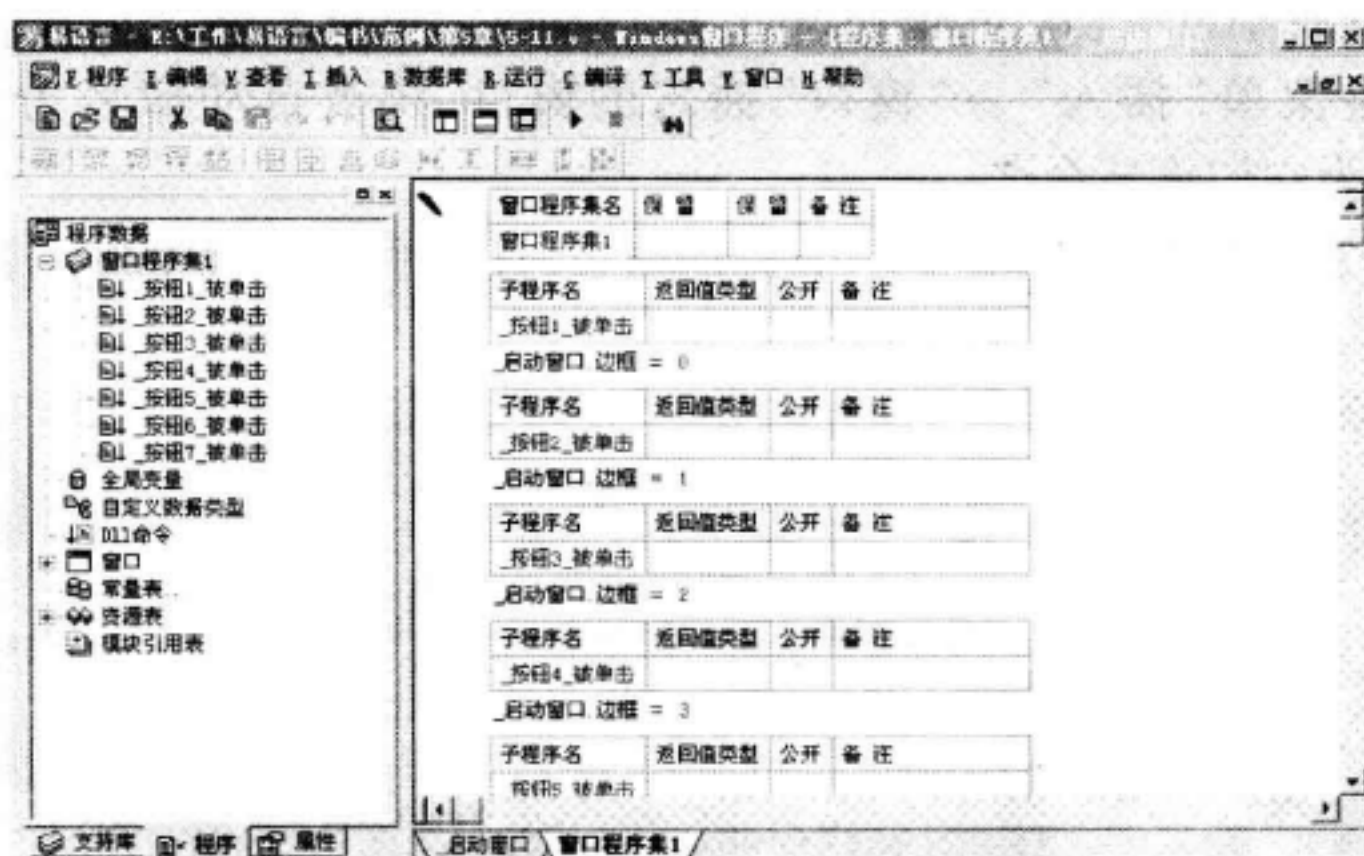



图5-12 分别为每个按钮输入程序

关闭程序设计界面,用鼠标双击第2个按钮,进入程序设计界面,并且重复上述动作,将每个按钮都输入相应的命令。

_启动窗口.边框 = 1

_启动窗口.边框 = 2

在窗口设置界面与程序设计界面中,可以用[Ctrl + Tab]组合热键进行切换。

现在命令可以试着运行一下这个程序,单击运行按钮,或选易语言系统主菜单“运行”→“运行”。运行后,就可以试着分别单击这几个按钮,看看窗口会发生如何的变化。

具体参见例程5-1。

运行例程5-2,结果如图5-13所示,试着分别单击窗口中的按钮,看看窗口的变化。再终止运行,回到程序设计状态,分别用鼠标双击按钮组件,看看其中的程序代码是如何控制窗口属性的。



图5-13 窗口属性测试运行结果

下面介绍窗体的其他属性。

1. “名称”属性

程序的第1个窗口必然是“_启动窗口”，它是由易语言自动生成的，它的“名称”必须是“_启动窗口”，编程者不可改动。其他的窗口、组件都可以设置名称。

通过易语言主菜单“插入”→“窗口”插入的窗口，将被自动命名为“窗口1”、“窗口2”……，编程者可为它们重新命名。

参见例程5-3，在本例程中除了一个启动窗口，还在程序面板中使用鼠标右键新建了4个新窗口，如图5-14所示。

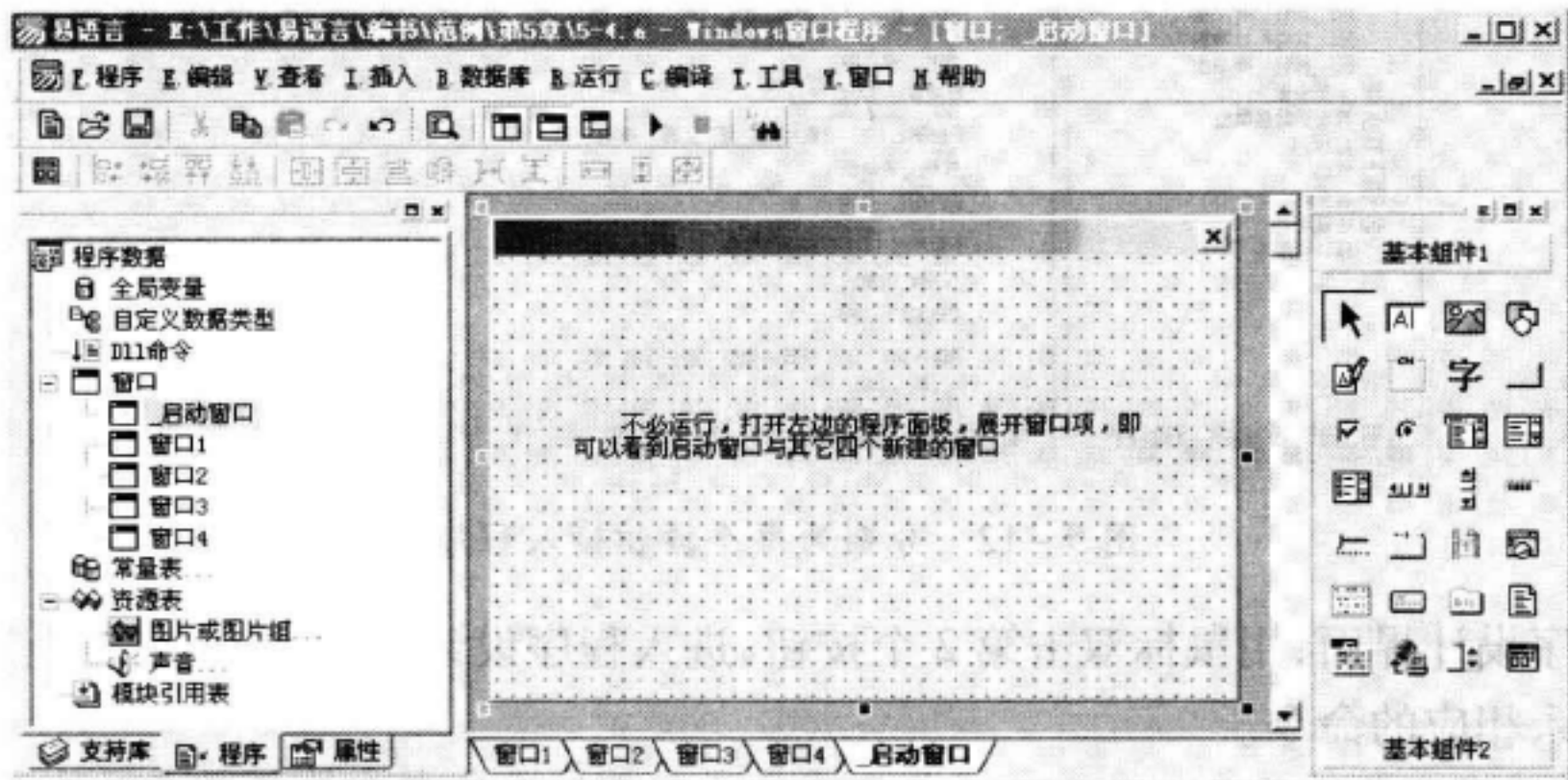


图 5-14 启动窗口与新窗口运行结果

易语言中的所有组件，除了“_启动窗口”外，编程者都可以为其重新命名。为组件取一个有意义的名称，是一个良好的编程习惯。一个好的名称 = 功能 + 组件类型，如“密码校验窗口”、“关于窗口”等。

参见例程5-4，在这个例程中将4个新建的窗口分别命名，并且在启动窗口中用4个按钮调用。如图5-15所示，在程序设计状态中，用鼠标双击这4个按钮组件，看看其中的程序代码是如何调用其他窗口的。

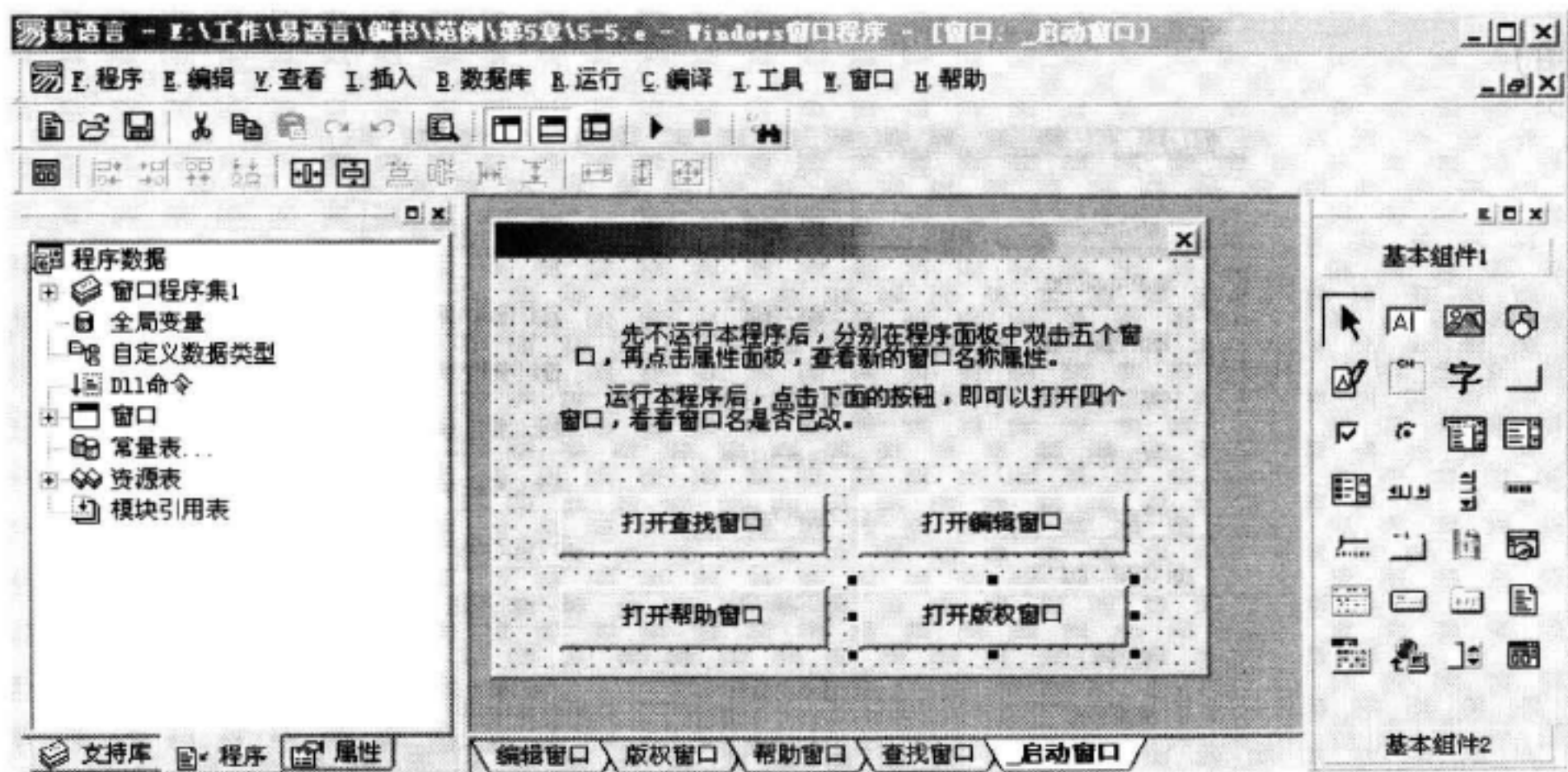


图 5-15 取窗口名称运行结果

其中一个按钮组件的程序代码如下：

载入(查找窗口,真)

2. “标题”属性

“标题”是显示于窗口标题栏上的文字,主要是给程序的最终使用者看的。

标题使用中文书写,可以将程序的名称、版本号等直接标在上面,也可以与窗口名称一致。使用“标题”属性后,可以看到窗口左上角显示出标题内容,参见例程 5-5,其运行结果如图 5-16 所示。

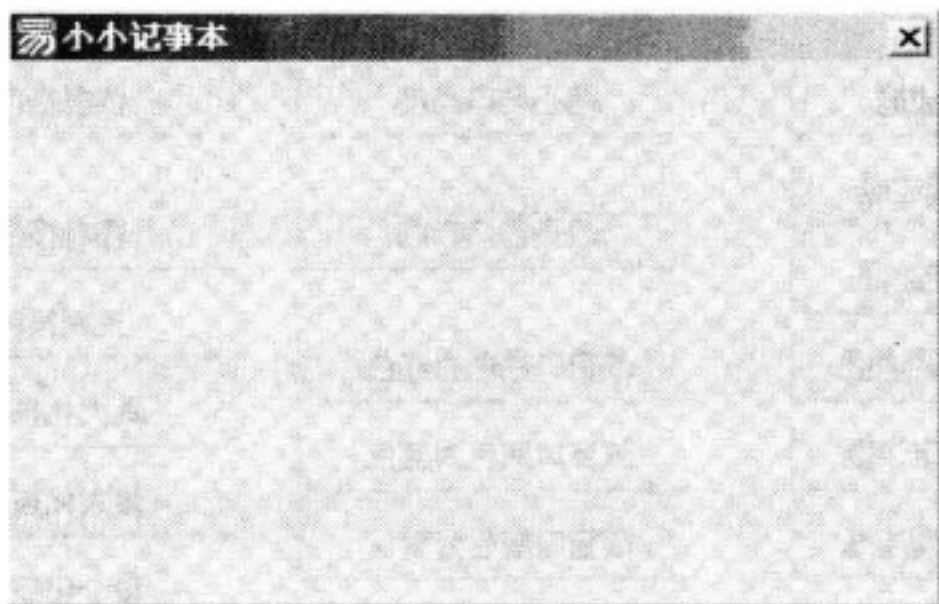



图 5-16 取窗口标题运行结果

也可以用代码在程序运行时为窗口改名称,程序代码如下：

_启动窗口.标题 = “小小记事本”

3. “边框”属性

窗口的“边框”属性有以下 7 种可选样项：“0. 无边框”、“1. 普通可调边框”、“2. 普通固定边框”、“3. 窄标题可调边框”、“4. 窄标题固定边框”、“5. 镜框式可调边框”、“6. 镜框式固定边框”等。默认值为“2. 普通固定边框”。

所谓“可调”表示窗口的大小可由程序使用者通过拖动鼠标调整。“固定”表示固定窗口的大小,不允许程序使用者调整。“窄标题”规定窗口的标题栏比常见的稍窄一些,且最左端没有图标、最右端只有 ;“镜框式”规定窗口的边框类似生活中的镜框式样。

因为窗口的边框属性默认为“2. 普通固定边框”,所以其大小不可调整——不可用鼠标拖动窗口边框的方法调整,如果单击最大化、最小化按钮,或者在代码中改变窗口的宽度和高度属性,窗口的大小还是会变的。有很多程序都使用“1. 普通可调边框”,让用户有调节窗口的能力。

参见例程 5-6,其运行结果如图 5-17 所示。

这个例程可以通过单击窗口中的 7 个按钮,随时改变窗口的“边框”属性。其中按钮改变窗口边框属性的程序代码如下。

_启动窗口.边框 = 0

_启动窗口.边框 = 1

4. “控制按钮”、“最大化按钮”、“最小化按钮”属性

三个逻辑型属性的值都只能为“真”或“假”。只有当“控制按钮”属性为“真”时,“最大化按钮”和“最小化按钮”才有意义。如果“控制按钮”为“真”,则“最大化按钮”和“最小化按钮”才有意义。

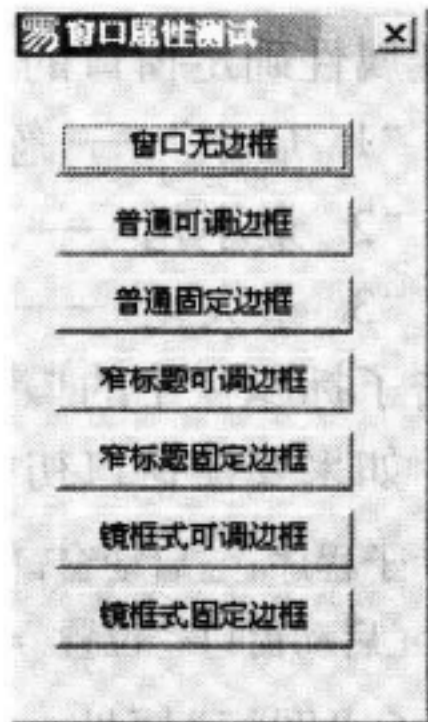







图 5-17 窗口属性测试运行结果 1

钮”分别控制窗口的标题栏右端是否出现  和  按钮。如果“控制按钮”为“假”，则窗口连标题栏也不显示了。

默认情况下，“控制按钮”的值为“真”，而“最大化按钮”和“最小化按钮”的值为“假”，所以窗口的标题栏上只显示  按钮，没有  和  按钮。

参见例程 5-7，运行结果如图 5-18 所示：其中，右侧的一竖排按钮可以控制“控制按钮”、“最大化按钮”、“最小化按钮”的属性。




图 5-18 窗口属性测试运行结果 2


5. “位置”属性

“位置”属性有 4 个可选值：

“0. 通常”——根据窗口的左边和顶边属性确定窗口在屏幕上的显示位置，根据宽度和高度属性确定窗口的大小；

“1. 居中”——忽略窗口的左边和顶边属性，直接把窗口显示于屏幕中央，这是默认值；

“2. 最小化”——最小化窗口到任务栏中（相当于单击了标题栏中的  按钮）；

“3. 最大化”——忽略窗口的左边、顶边、宽度和高度属性，以全屏方式显示窗口（相当于单击了标题栏中的  按钮）。

如果要使窗口初始状态为最大化，可以使用如下的程序代码：

```
子程序：_启动窗口_创建完毕
_启动窗口.位置 = 3
```

6. “图标”属性

“图标”属性用于指定显示于窗口标题栏最左端的图标。目前只支持 16 × 16 像素、32 × 32 像素 16 色的图标。

如果没有为整个程序设置图标（通过易语言主菜单“工具”→“程序配置”），则会以“_启动窗口”的“图标”属性指定的图标作为最后生成的可执行文件的图标。如果没有给“_启动窗口”设置“图标”属性，就会以易语言默认的图标作为程序的图标。

7. “底色”属性

“底色”属性用于指定窗口的背景颜色。通常窗口底色是浅灰色的（默认底色）。当窗口

有底图时,本属性无效。

8. “底图”、“底图方式”属性

“底图”属性用于指定窗口的背景图片。如果设置了该属性,此时窗口的“底色”属性不再生效,还可以用“底图方式”属性来控制图片的显示方式。

“底图方式”的可选值有:“0. 图片居左上”,“1. 图片平铺、”,“2. 图片居中”,默认值为“1. 图片平铺”。

9. “背景音乐”、“播放次数”属性

“背景音乐”属性可以为窗口指定背景音乐,在窗口显示后即开始播放。支持 WAV、MIDI 格式的音乐文件。

如果设置了该属性,还可以通过“播放次数”属性来控制音乐的播放次数:“0. 循环播放”、“1. 仅播放一次”、“2. 不播放”,默认值为“0. 循环播放”。

10. “Esc 键关闭”属性

如果其值为真,可以通过按键盘上的 Esc 键关闭本窗口;如果其值为“假”,则 Esc 键无效,默认值为“真”。

11. “可否移动”属性

如果其值为“真”,可以通过拖动窗口的标题栏来调整窗口的位置;如果其值为“假”,则不可以,默认值为“真”。

12. “随意移动”属性

如果其值为“真”,可以通过拖动窗口任意位置来调整窗口的位置;如果其值为“假”,则不可以。默认值为“假”。

自己动手试一试,将上述属性改变后,用鼠标单击窗口,看看是否能移动。

再来做个小小的测试:

为“_启动窗口”添加一个按钮组、一个图形按钮组件和一个标签组件,并分别为按钮组件、图形按钮组件和标签组件加入“鼠标左键被按下”和“鼠标左键被放开”事件并添加代码(如弹出信息框)。


当窗口的“随意移动”属性为“假”的时候,3 种组件都可以接受“鼠标左键被按下”和“鼠标左键被放开”事件。

当窗口的“随意移动”属性为“真”的时候,按钮组件因为可以接受焦点,所以可以接受“鼠标左键被按下”和“鼠标左键被放开”事件;标签组件因为不可以接受焦点,所以不接受“鼠标左键被按下”和“鼠标左键被放开”事件;图形按钮组件却只能接受“鼠标左键被放开”事件而不接受“鼠标左键被按下”事件。

13. “总在最前”属性

如果其值为“真”,窗口永远显示在屏幕的最前面,默认值为“假”。除此之外,窗口还有其他一些属性,这里不再一一介绍。

5.1.3 窗口的事件

前述只是对窗口的“外形”、“尺寸”以及“边框”等属性进行了设置,虽然可以在编程中也进行一系列的控制,但是否可以像按钮组件那样,单击或关闭时产生一些动作呢?这就需要知道“事件”这个概念。

“事件”是可视化编程的一种编程模式,就是当组件被改变时,会产生的一系列操作,我们

把它称为“事件”。

1. 认识窗口事件

每个组件都在属性面板下方有一个下拉列表,如图 5-19 所示。

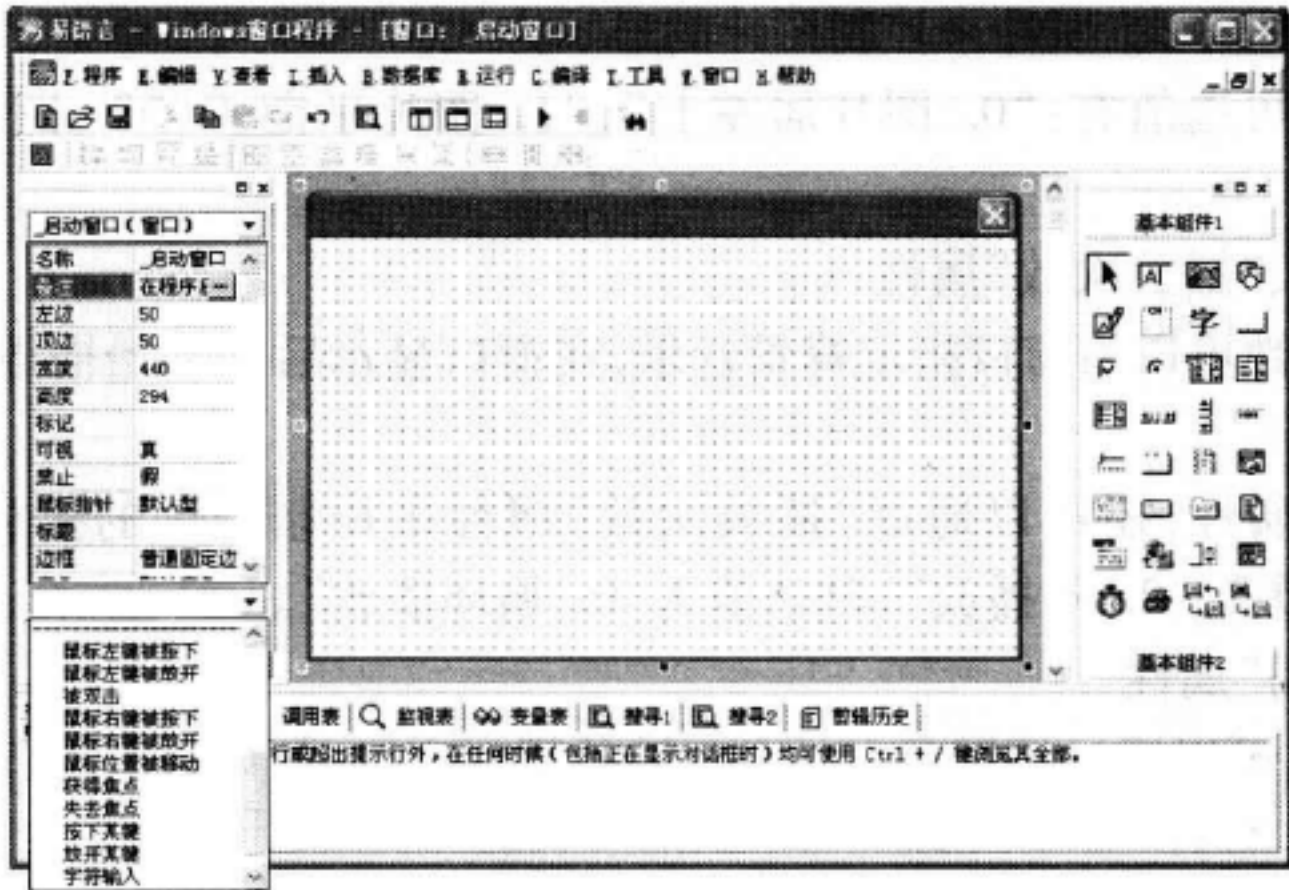


图 5-19 窗口事件下拉列表

窗口的事件有鼠标左键被按下、鼠标左键被放开、被双击、鼠标右键被按下、鼠标右键被放开、鼠标位置被移动、获得焦点、按下某键、放开某键、字符输入、创建完毕、位置被改变、尺寸被改变、将被销毁、可否被关闭、托盘事件、被激活、首次激活、被取消激活、空闲、被显示、被隐藏等。

2. 鼠标单击事件例程

下面,介绍窗口事件激活的小例程。

新建一个程序,出现一个空白的启动窗口。单击属性面板下方的事件下拉菜单,选中“鼠标左键被按下”事件,这时会进入“__启动窗口_鼠标左键被按下”的程序编辑界面,如图 5-20 所示。

窗口程序集名	保留	保留	备注
窗口程序集1			

子程序名	返回值类型	公开	备注		
__启动窗口_鼠标左键被按下	逻辑型				
参数名	类型	参考	可空	数组	备注
横向位置	整数型				
纵向位置	整数型				
功能键状态	整数型				
_启动窗口.标题 = “这是鼠标左键被按下”					

子程序名	返回值类型	公开	备注		
__启动窗口_鼠标右键被按下	逻辑型				
参数名	类型	参考	可空	数组	备注
横向位置	整数型				
纵向位置	整数型				
功能键状态	整数型				
信息框 (“这是鼠标右键被按下”, 0,)					

图 5-20 输入程序代码(1)

在产生了“鼠标左键被按下”事件中输入以下程序代码:

_启动窗口.标题 = “这是鼠标左键被按下”

在产生了“鼠标右键被按下”事件中输入以下程序代码:

信息框(“这是鼠标右键被按下”,0,)

试运行这个程序,当单击鼠标左键时,会在窗口标题上显示一排文字:“这是鼠标左键被按下”,当单击鼠标右键时,会弹出一个信息框,显示:“这是鼠标右键被按下”。

当然,这是一个非常简单的例子,而在实际的编程中,可以根据需要在事件下扩充相应的代码。情况会稍微复杂一些,鼠标单击后所产生的程序代码是不一样的。

具体参见例程 5-8。

3. 创建完毕事件例程

重新建立一个新程序,产生一个新的空白的启动窗口。单击属性面板下方的事件下拉菜单,选中“创建完毕”事件,这时会进入“__启动窗口_创建完毕”的事件子程序编辑界面,如图 5-21 所示。

在“__启动窗口_创建完毕”的事件子程序下方输入一行代码:

_启动窗口.标题 = “窗口创建完毕,开始运行!”

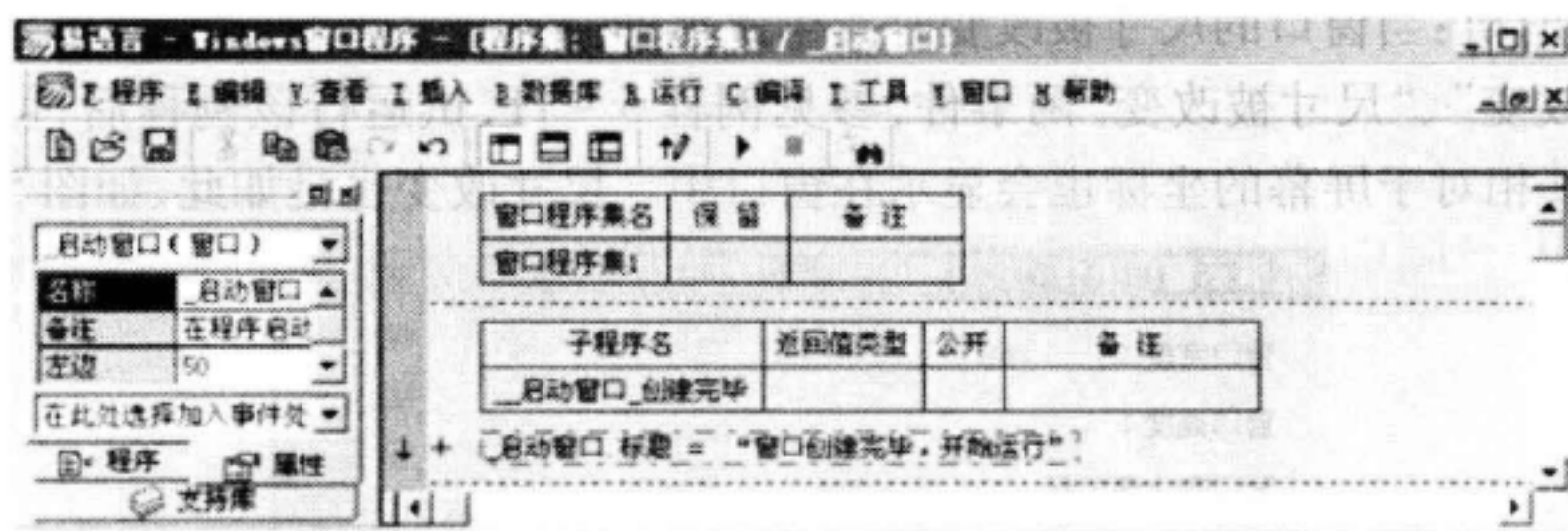


图 5-21 输入程序代码(2)

试运行该程序,窗口的标题开始是没有任何文字的,但当窗口被创建时,才由“__启动窗口_创建完毕”的事件为窗口写上标题文字:“窗口创建完毕,开始运行!”。

具体参见例程 5-9。

4. 窗口大小被改变事件例程

再建立一个新的程序,产生一个新的空白的启动窗口。

单击属性面板中的“边框”属性,将此属性调整为:“普通可调边框”,必须进行该操作,否则程序运行时将无效果。

单击属性面板下方的事件下拉菜单,选中“尺寸被改变”事件,这时会进入“__启动窗口__尺寸被改变”的程序编辑界面,如图 5-22 所示。

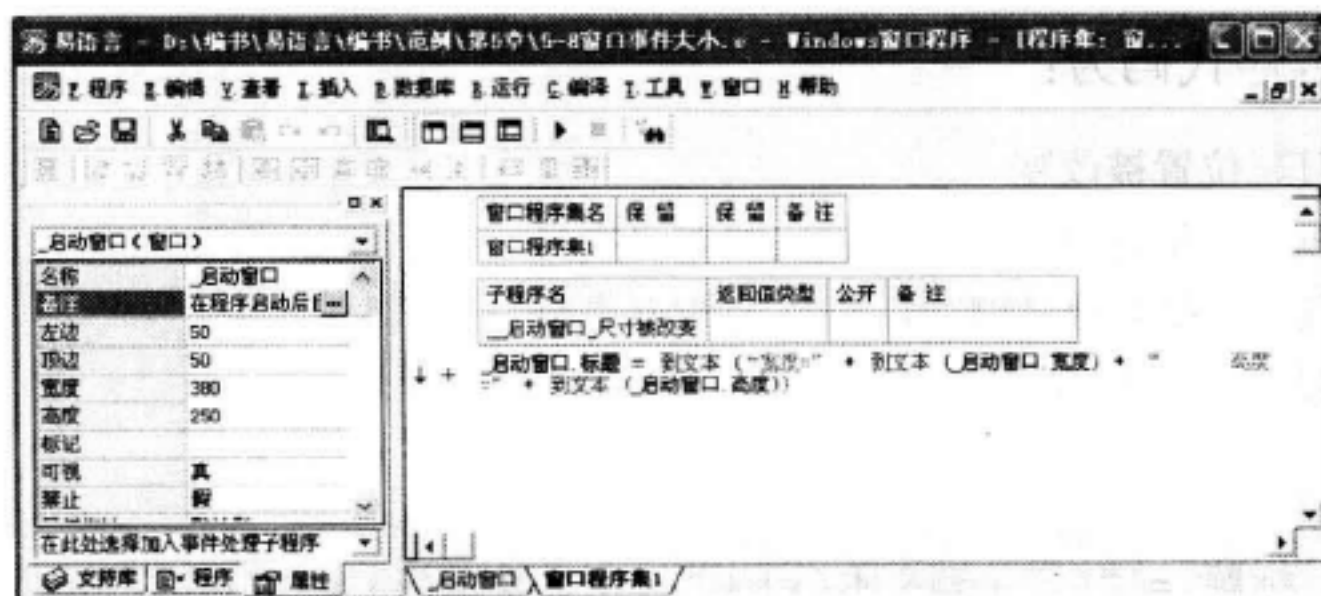


图 5-22 输入程序代码(3)

输入以下程序代码：

```
_启动窗口.标题 = "宽度=" + 到文本(_启动窗口.宽度) + " 高度=" + 到文本(_启动窗口.高度)
```

试运行程序,这个窗口是可以随意改变尺寸的。当改变尺寸后,在窗口的标题上就会显示不同的尺寸。具体参见例程 5-10。

5. “创建完毕”事件

事件产生时机:当窗口及其中的所有组件均被创建后在显示之前产生此事件。

“创建完毕”事件是窗口最重要的事件,也是最常用的事件。通常在本事件的“事件处理子程序”中做一些关于本窗口的初始化工作。“_启动窗口”的“创建完毕”事件尤其重要,通常在里面做整个程序的初始化工作。

6. “位置被改变”事件

事件产生时机:当窗口的位置被改变时产生此事件。

7. “尺寸被改变”事件

事件产生时机:当窗口的尺寸被改变时产生此事件。

“位置被改变”、“尺寸被改变”两事件,参见例程 5-11,试运行该例程后,可以看到当窗口被移动后,其相对于屏幕的坐标也会显示在窗口中。尺寸改变也是如此,如图 5-23 所示。

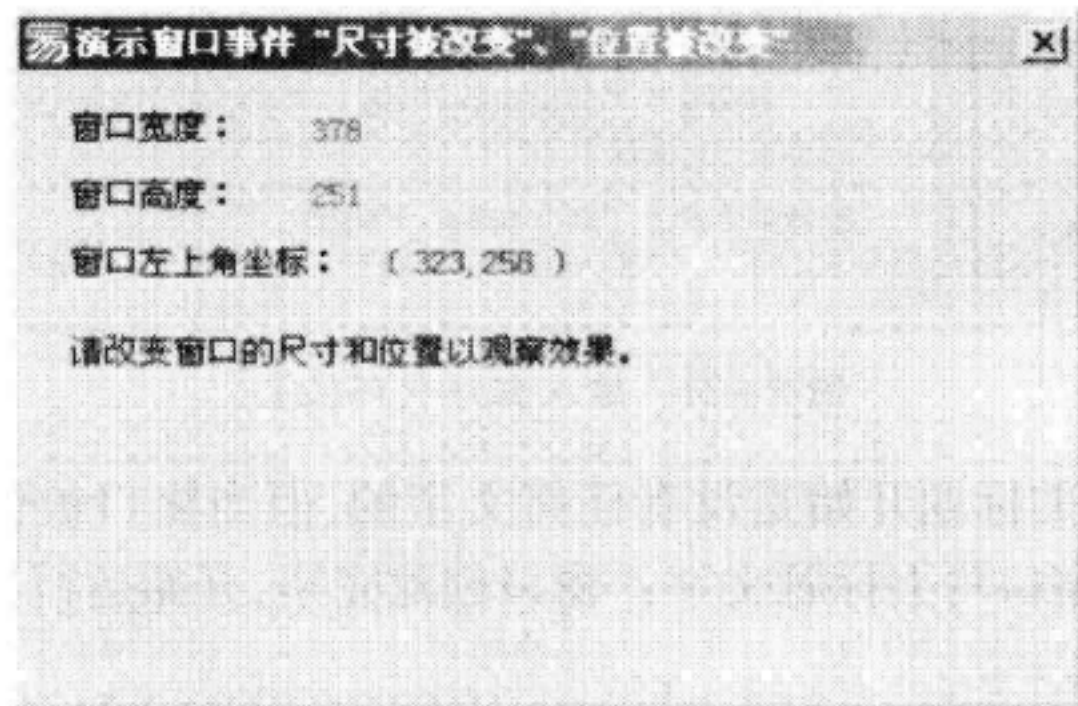


图 5-23 窗口尺寸位置被改变运行结果

其中取窗口宽度与高度的程序代码为：

子程序: _启动窗口_尺寸被改变

宽度标签.标题 = 到文本(_启动窗口.宽度)

高度标签.标题 = 到文本(_启动窗口.高度)

取窗口坐标的程序代码为：

子程序: _启动窗口_位置被改变

局部变量:窗口左边 数据类型:整数型 备注:自定义变量

局部变量:窗口顶边 数据类型:整数型 备注:自定义变量

窗口左边 = _启动窗口.左边

窗口顶边 = _启动窗口.顶边

左上角坐标标签.标题 = “(” + 到文本(x) + “,” + 到文本(y) + “)”

增加两个标签,再试一试下面的程序代码：


__启动窗口_鼠标位置被移动

宽度标签 1. 标题 = 到文本 (取鼠标水平位置 ())

高度标签 1. 标题 = 到文本 (取鼠标垂直位置 ())

8. “将被销毁”事件

事件产生时机:当窗口被销毁之前产生此事件。

用户单击窗口右上角的  按钮,或执行到命令代码“销毁()”时,就会自动产生此事件。如果有本事件的处理代码,则先执行它们,然后销毁窗口,否则直接销毁窗口。

程序员通常在本事件的“事件处理子程序”中做一些收尾工作,例如检查有没有存盘,如果没存盘,用对话框提示存盘等。

具体参见例程 5-12。此例程运行后,在关闭程序之前,运行了如下程序代码:

子程序: __启动窗口_将被销毁


信息框 (“感谢您使用本程序!”, 0,)

上述程序表示在关闭程序时,打开一个信息框,显示“感谢您使用本程序!”,如图 5-24 所示。单击对话框中的“确定”按钮后就结束运行。

9. “可否被关闭”事件

事件产生时机:在窗口关闭之前产生此事件,稍早于“将被销毁”事件。

本事件有一个逻辑型返回值。如果在在本事件的“事件处理子程序”中返回“假”,表示不允许关闭窗口;如果返回“真”或不返回值,表示允许关闭窗口。(窗口要先关闭,即从屏幕上消失,再销毁,即从内存中消失,所以“可否被关闭”稍早于“将被销毁”产生。)

有时候,程序的使用者不小心按了窗口右上角  关闭按钮,可能立刻就后悔了;而且有时候程序可能因为某种原因不允许关闭窗口。这就是响应“可否被关闭”事件的时候了。

参见例程 5-13,运行结果如图 5-25 所示。



图 5-24 程序运行结果

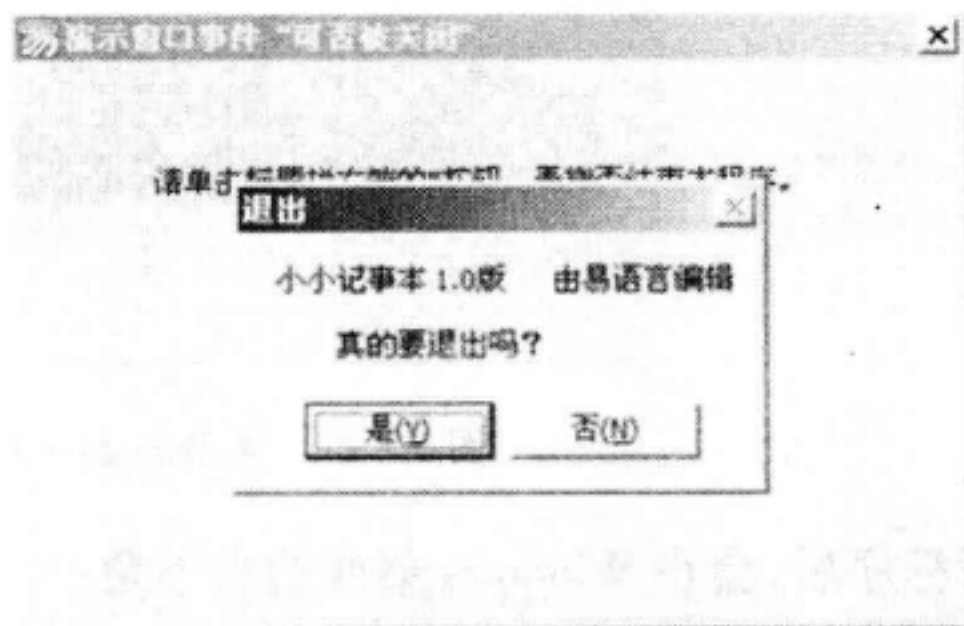


图 5-25 可否被关闭事件运行结果

试运行这个程序,当要关闭这个窗口时,就会有一个提示信息框出现,要求确认是否直接的要退出。

在此事件中使用的代码如图 5-26 所示。

10. “托盘事件”事件

事件产生时机:当用鼠标单击或双击任务栏托盘中的图标(用“置托盘图标()”命令设置)时产生本事件。

托盘事件有一个整数型参数“操作类型”,其值可能是:“1. #左键单击”;“2. #左键双击”;

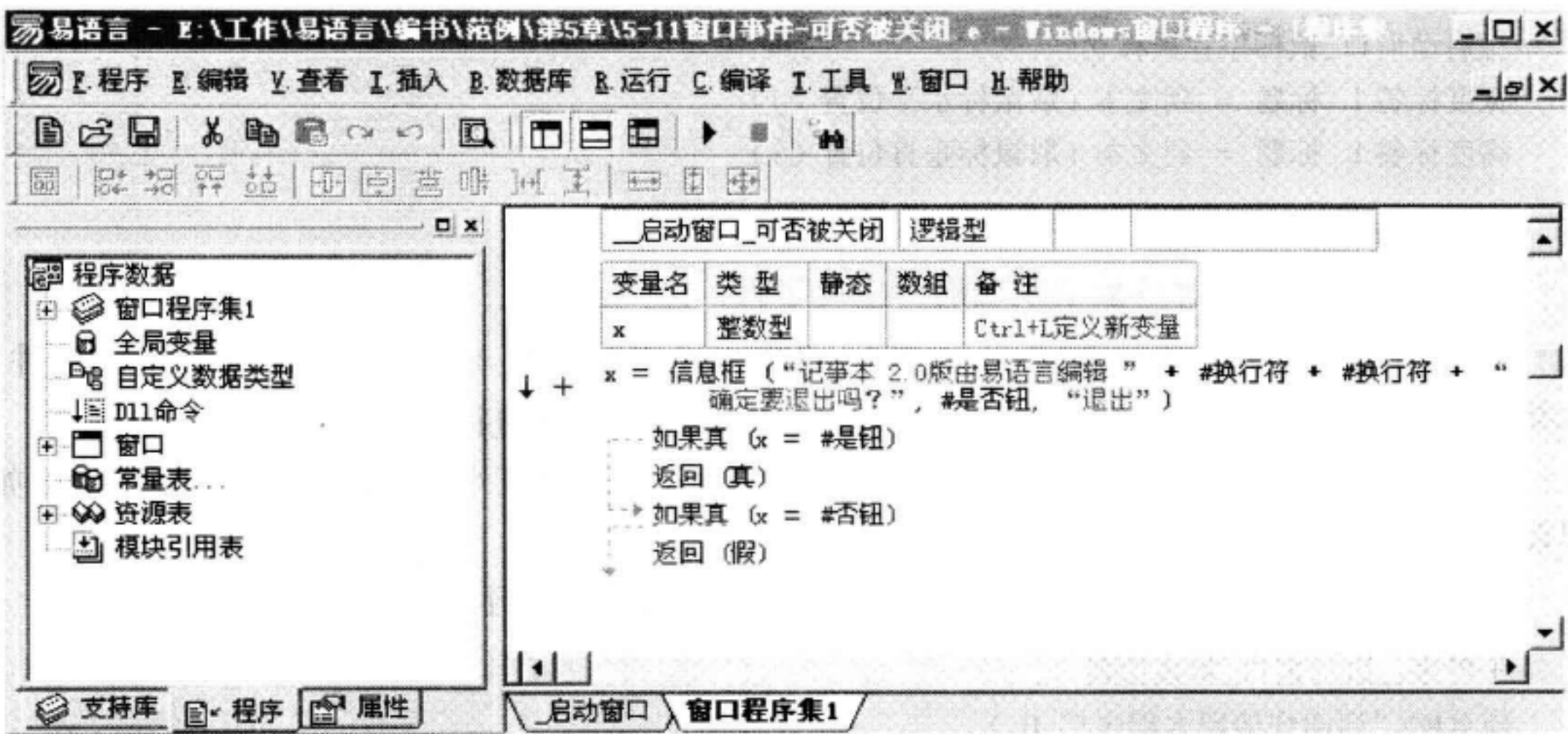


图 5-26 可否关闭事件程序代码

“3. #右键单击”。程序员可根据此参数判断出用户的击键动作,再做相应处理,比如是弹出菜单还是显示窗口等。

例如,下载软件“网络蚂蚁”的托盘图标,双击时显示主窗口,右击弹出托盘菜单,左击则没有反应。参见例程 5-14,运行结果如图 5-27 所示。

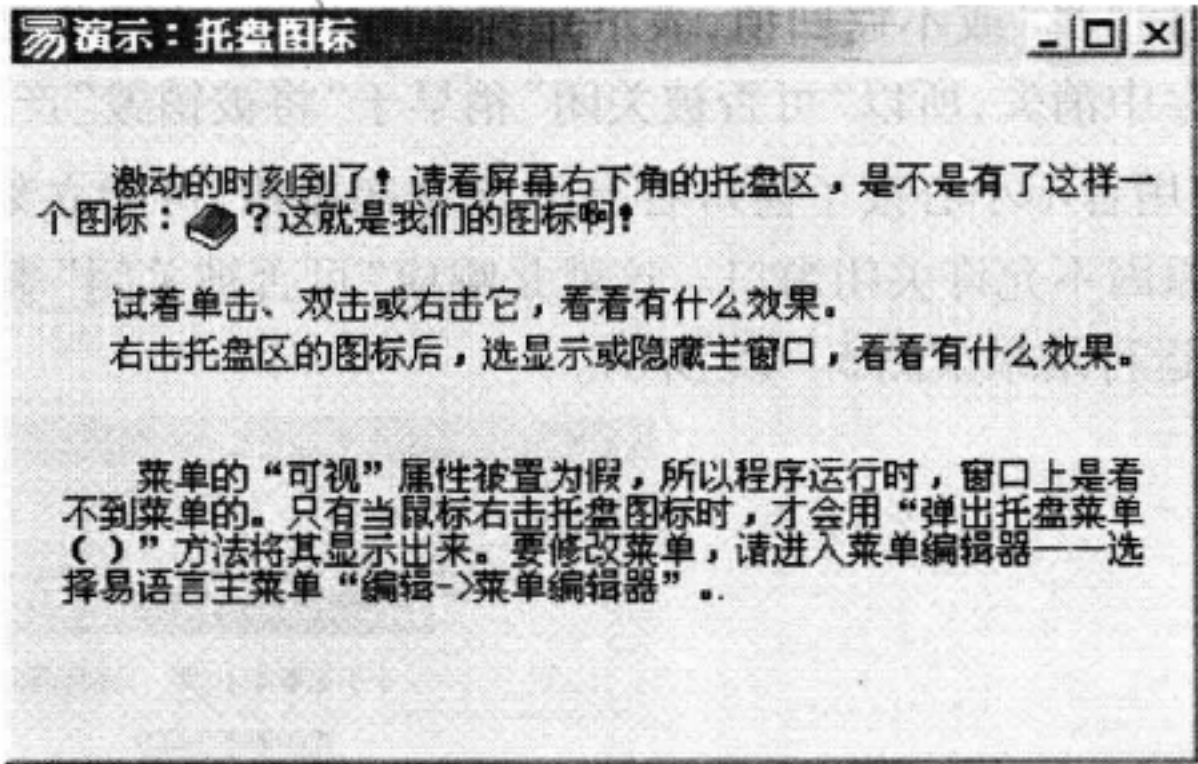


图 5-27 系统托盘 - 鼠标点击运行结果

运行程序后,会在 Windows 桌面的右下角——系统托盘中生成一个小图标。当用鼠标右击时,会弹出一个菜单,选“显示主窗口”或“隐藏主窗口”选项,都可以观察操作结果。

图标被导入图片资源中,并且在程序代码中引用,程序代码如下:

子程序:__启动窗口_创建完毕

置托盘图标(#图标 1,“记事本工具”)

如何导入图片资源及更多的托盘事件,将在后面内容中做详细介绍。

11. “被激活”、“首次激活”、“被取消激活”事件

事件产生时机:当窗口被激活、首次激活、被取消激活时产生本事件。

窗口的标题栏由灰色变为蓝色时,表示窗口“被激活”;由蓝色变为灰色时,表示“被取消激活”;“首次激活”即窗口被显示后第一次激活。“首次激活”事件在窗口被销毁之前只会产

生一次,而“被激活”、“被取消激活”事件可能产生很多次。

参见例程 5-15,例程的程序运行后如图 5-28 所示。当启动窗口初次被激活时,会弹出一个信息框。当被激活时,就会播放音乐,如果被取消激活,就会鸣叫。

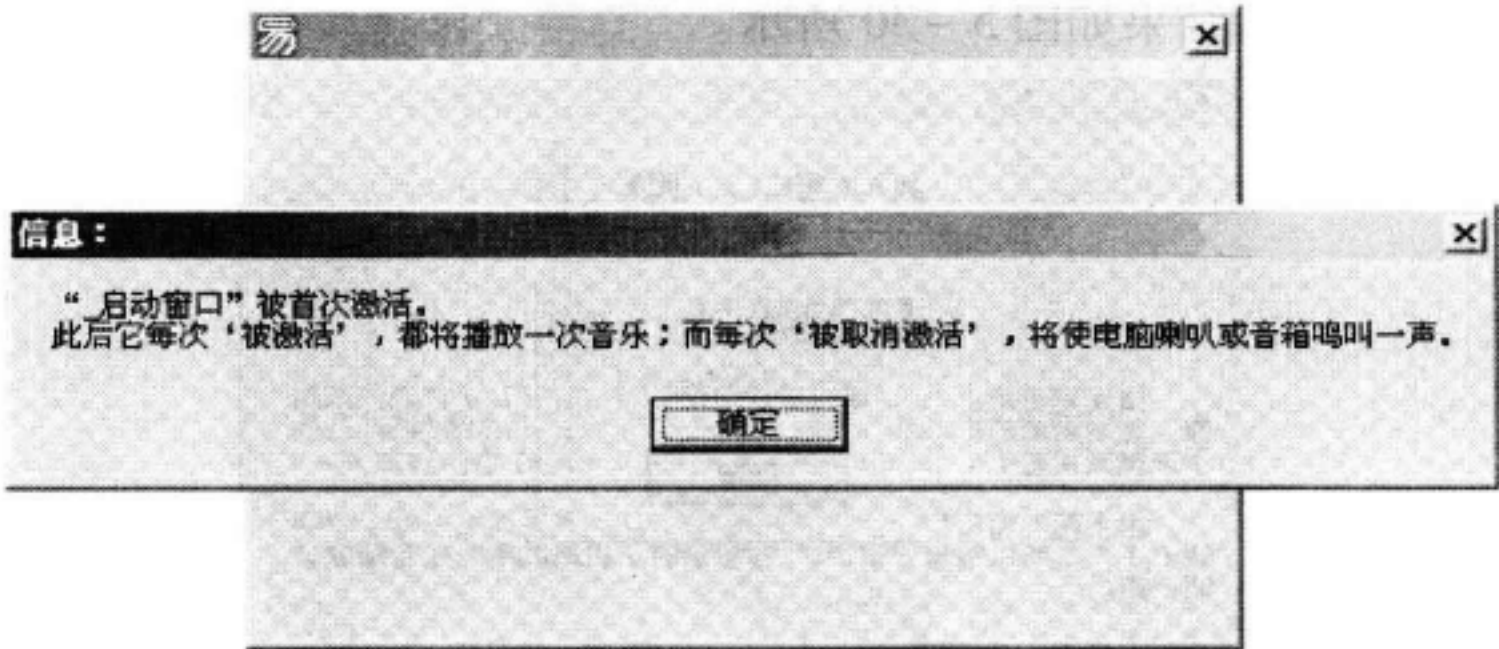


图 5-28 启动窗口初次被激活运行结果

参见例程 5-16,程序运行后,单击最上面一个按钮,可以弹出一个新的“窗口 1”(如图 5-29 所示),当没有激活这个窗口 1 时,它没有任何动作。当通过启动窗口的第二个按钮激活它,或者直接单击窗口 1,它就会产生鸣叫。

本例程中有两个窗口:“_启动窗口”与“窗口 1”。“窗口 1”由“_启动窗口”的“载入()”命令弹出的。

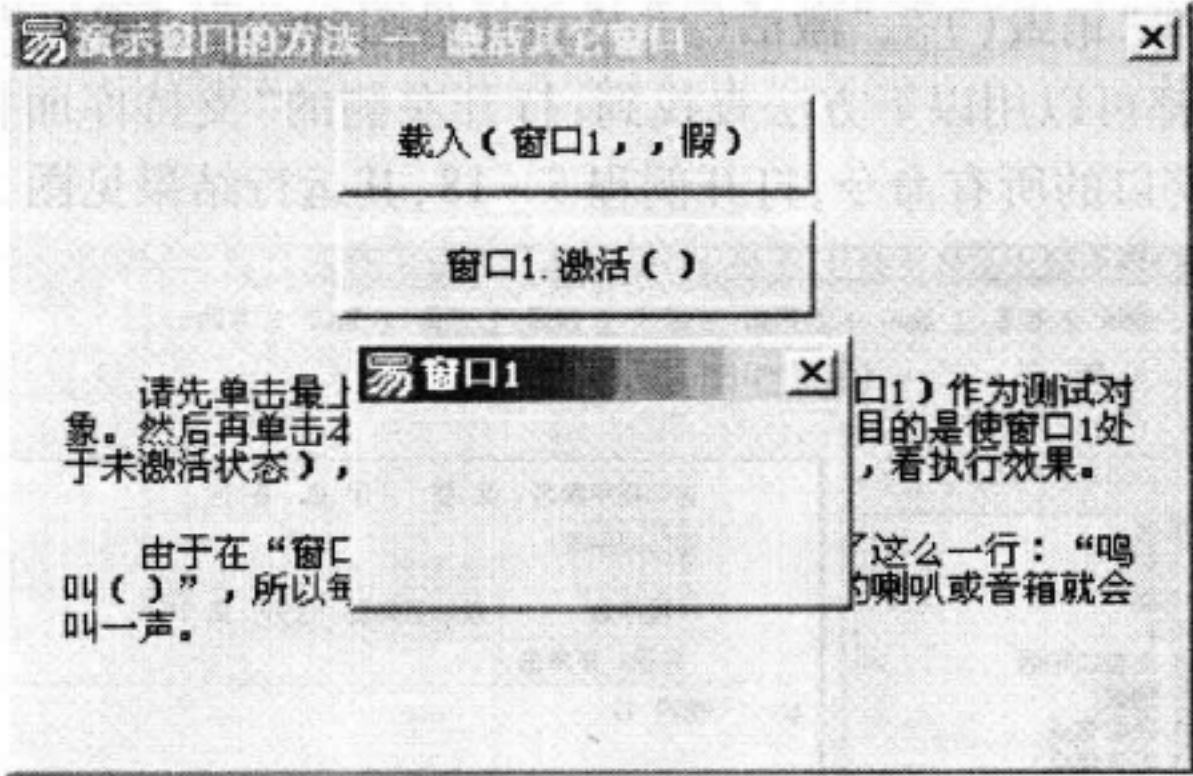


图 5-29 单击按钮运行结果

12. “空闲”事件

事件产生时机:当系统(CPU、操作系统或程序)空闲时产生本事件。

本事件有一个整数型参数“已空闲时间”,用于告知编程者事件产生时,系统已经进入空闲状态的时间,单位为毫秒。用户程序在空闲状态下所进行的大工作量操作最好等到此值较大时进行。

本事件有一个逻辑型的返回值。如果事件处理子程序返回“假”或者不返回值,在此次空闲期间系统将不再产生空闲事件(可以降低 CPU 占用率)。如果返回“真”,系统将产生空闲事件。

“空闲”即系统(CPU 或操作系统或程序)“有空了、清闲了”。通常在“空闲”事件的“事件处理子程序”中做一些后台处理工作。本事件不常用。

13. “被显示”、“被隐藏”事件

事件产生时机:当窗口被显示、被隐藏(比如令其可视属性为假)时产生本事件。
“被显示”事件在窗口被销毁之前至少产生一次;“被隐藏”事件可能一次也不产生。
参见例程 5-17,运行结果如图 5-30 所示。

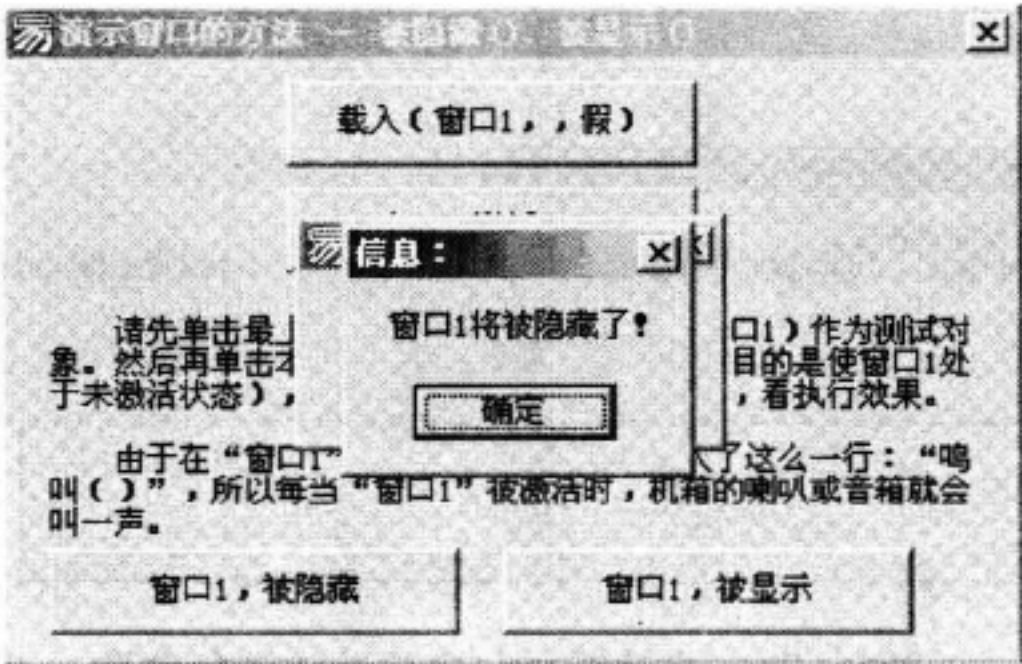


图 5-30 隐藏与显示事件运行结果

运行中,每次在“窗口 1”被隐藏之前都有提示信息,在实际编程中,可以需要修改并应用它。

5.1.4 窗口的方法

窗口的重要方法有“销毁()”、“激活()”、“置托盘图标()”、“弹出托盘菜单()”等。
窗口的所有方法都可以用以下方法查找到:打开左侧的“支持库面板”,展开“数据类型”→“窗口”中,列出了窗口的所有命令,打开例程 5-18,其运行结果见图 5-31。

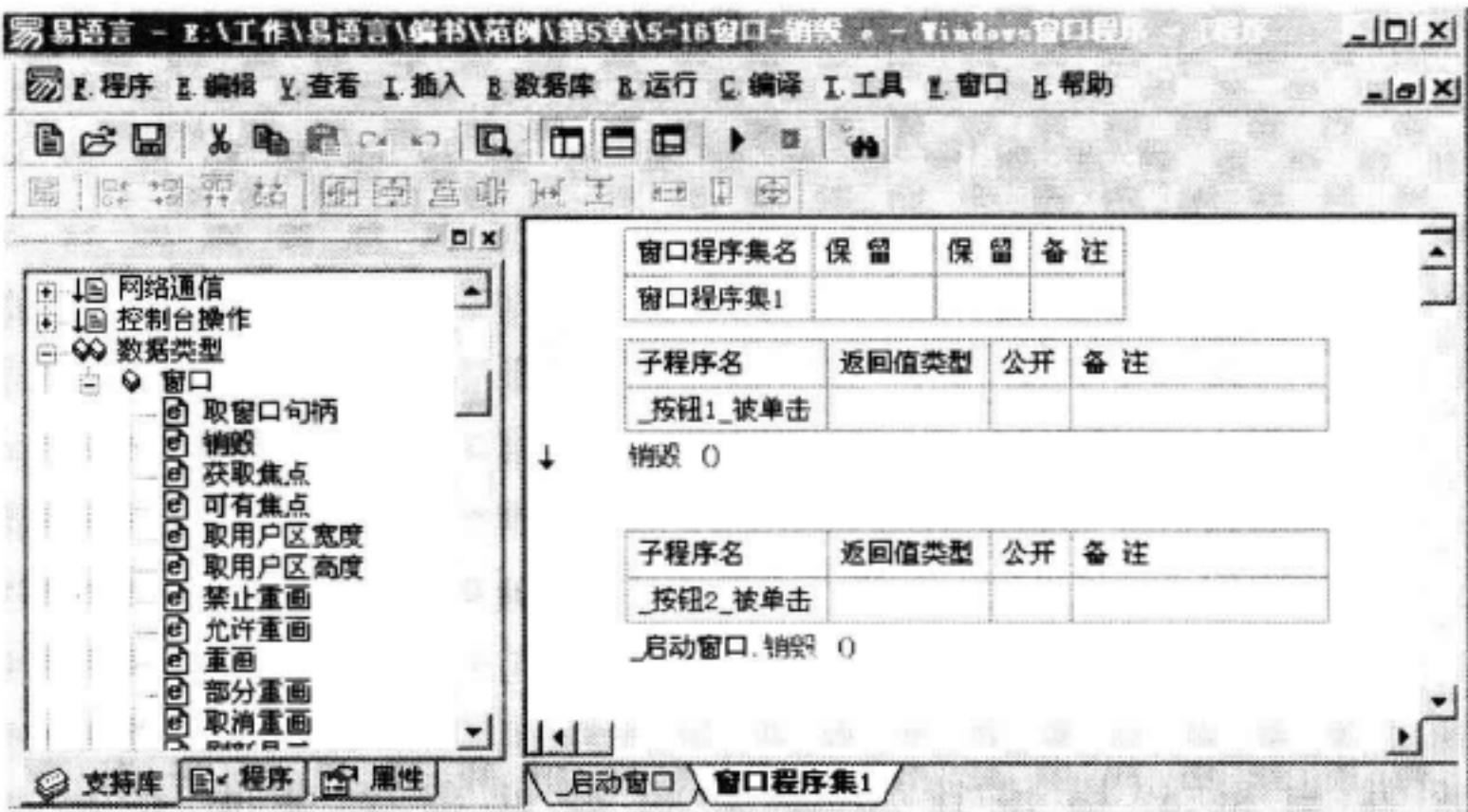


图 5-31 支持库面板中的窗口方法

1. “移动()”方法

光标放到“移动()”命令上后,使用 F1 帮助键,可以查看到这个命令的帮助文件,如图 5-32 所示。

图 5-32 中的“移动()”是一个系统提供的命令。从帮助文件里面可以看出,这个命令前面要引用一个对象,该命令写法如下:

_启动窗口.移动(,,)

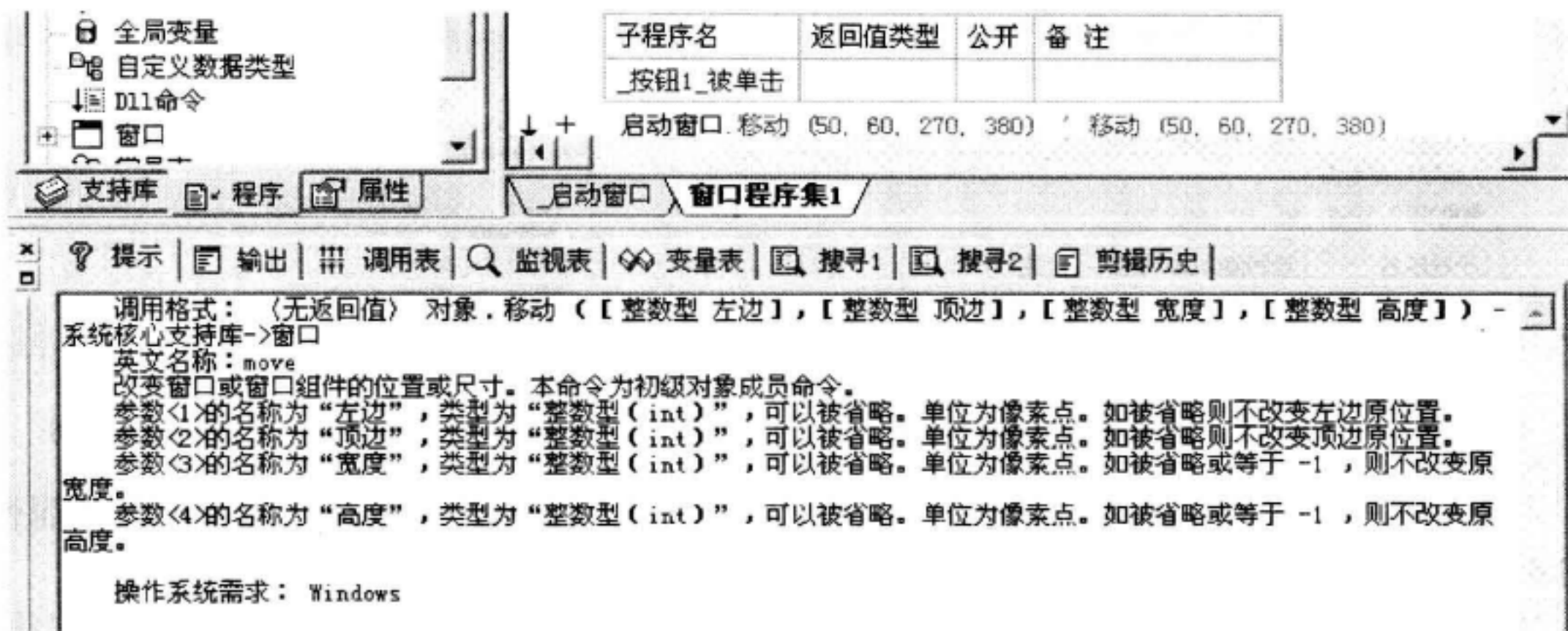


图 5-32 帮助文件

默认的情况下可以不写对象,但会移动当前的窗口。后面可以跟 4 个参数,4 个参数分别代表移动到目的地的坐标和移动后窗口尺寸属性。

请试一下:新建一个窗口,放一个按钮组件,双击按钮组件后,进入程序设计界面,输入以下命令。

```
_启动窗口.移动(50,60,270,380)
```

如果输入以下程序代码,运行效果是一样的:

```
移动(50,60,270,380)
```

具体参见例程 5-18。

参见例程 5-19,该例程由两个窗口组成,可由启动窗口控制帮助窗口的移动,也可以由帮助窗口控制启动窗口的移动。注意在弹出“窗口 1”时的命令中,第 3 个参数要设置为“假”:

```
载入(帮助窗口,,假)
```

这样可以同时存在两个窗口。然后只要分别在启动窗口建立两个新按钮,分别输入以下命令,如图 5-33 所示。

```
子程序:_按钮 2_被单击  
移动(50, 60, 270, 380)
```

```
子程序:_按钮 3_被单击  
帮助窗口.移动(150, 260, 270, 380)
```

在帮助窗口新建两个按钮,分别输入以下命令,如图 5-34 所示。

```
子程序:_按钮 1_被单击  
帮助窗口.移动(40, 50, 60, 70)
```

```
子程序:_按钮 2_被单击  
_启动窗口.移动(40, 150, 360, 470)
```

窗口程序集名	保留	保留	备注
窗口程序集2			
子程序名	返回值类型	公开	备注
_按钮1_被单击			
帮助窗口.移动 (40, 50, 60, 70)			
子程序名	返回值类型	公开	备注
_按钮2_被单击			
+ _启动窗口.移动 (40, 150, 360, 470)			

图 5-33 为启动窗口的两个新按钮输入命令

窗口程序集名	保留	保留	备注
窗口程序集1			
子程序名	返回值类型	公开	备注
_按钮1_被单击			
载入(帮助窗口, 假)			
子程序名	返回值类型	公开	备注
_按钮2_被单击			
移动 (50, 60, 270, 380)			
子程序名	返回值类型	公开	备注
_按钮3_被单击			
帮助窗口.移动 (150, 250, 270, 380)			

图 5-34 为帮助窗口的两个新按钮输入命令

本例程中的一行程序“_启动窗口.移动(50, 60, 270, 380)”也可以用以下的改变属性的 4 行程序实现,但有本质的区别。

_启动窗口.左边 = 50
_启动窗口.顶边 = 60
_启动窗口.宽度 = 270
_启动窗口.高度 = 380

这两个方法的区别在于,一种方法命令,另一种方法调用的是属性。

2. “销毁()”方法

语法:窗口名称.销毁()

既然可以载入一个窗口,那么也可以将这个窗口去除,虽然可以单击窗口右上角的×来销毁并关闭窗口,但在程序中如何实现呢?使用“销毁()”命令即可,如图 5-35 所示。

_启动窗口.销毁()

如果没有指明窗口对象,只有“销毁()”命令,会将当前激活的窗口作为销毁对象。如果当前为主启动窗口,被销毁后,程序也就自然退出了。

“销毁()”命令与“结束()”命令不同,“结束()”命令不仅销毁当前窗口(当前窗口不必是启动窗口)也随时终止了程序的运行。

窗口程序集名	保留	保留	备注
窗口程序集1			
子程序名	返回值类型	公开	备注
_按钮1_被单击			
↓ 销毁 0			
子程序名	返回值类型	公开	备注
_按钮2_被单击			
_启动窗口.销毁 0			

图 5-35 输入程序代码

本例中,在一个新建的窗口中建立两个按钮,分别是“按钮 1”与“按钮 2”,分别双击这两个按钮,分别输入“销毁()”与“_启动窗口.销毁()”来测试,这两个命令的运行结果是一样的,如图 5-27 所示。具体参见例程 5-20。

3. “激活()”方法

功能:激活窗口——如果原来标题栏是灰色则变为蓝色。

语法:窗口名称.激活()

应用实例,在事件中输入以下程序代码:

_启动窗口. 激活()

将激活“_启动窗口”,输入以下程序代码:

窗口 1. 激活()

将激活“窗口 1”,如果在窗口未激活状态下调用本命令,将会激活窗口,并产生“被激活”事件;如果在窗口激活状态下调用本命令,不会产生“被激活”事件。具体参见例程 5-21,使用的程序代码如下:

窗口 1. 激活()

在前面介绍“被激活”事件时已接触到这个方法,前面介绍的是一个事件,是在窗口 1 中产生一个“_窗口 1_被激活”事件子程序,其中的程序代码是:

鸣叫()

表示当被激活时,就鸣叫一次。所以,两者又是有区别的。

单击启动窗口中的激活按钮,产生了一个命令:“窗口 1. 激活()”,而此命令激活了窗口 1,窗口 1 又产生了一个事件:“_窗口 1_被激活”,而此事件又激活了一个命令:“鸣叫()”,因此请务必理解上述概念。

4. “置托盘图标()”方法

功能:设置本程序在系统托盘中的图标,如图 5-36 所示。

语法:窗口名称. 置托盘图标([图标数据],[提示信息])

图标数据——指定出现于托盘中的图标,其值可为图标资源、图标字节集数据或图标文件名,如果省略本参数,表示清除已经设置了的托盘图标。

提示信息——当把鼠标放到托盘图标上时,出现的提示文本。

本命令通常在“_启动窗口”的“创建完毕”事件中设置托盘图标,在“_启动窗口”的“将被销毁”事件中以空参数(如:置托盘图标(,))清除已经设置了的托盘图标。即使设置了托盘图标后,退出程序前没有清除,易语言将自动清除托盘图标。

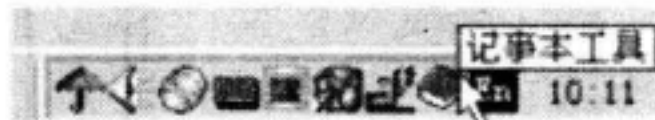


图 5-36 系统托盘

应用实例:

置托盘图标(#图标 1,“记事本工具”)

上述程序代码表示:在程序面板中的资源中,有一个名为“图标 1”的图标图片资源,并且作为图标显示在系统托盘中。提示行为“记事本工具”。

置托盘图标(图片框 1. 图片,“记事本工具”)

上述程序代码表示:窗口中还有一个名为“图片框 1”的组件,其中有一个图标图片,并且作为图标显示在系统托盘中,提示行为“记事本工具”。

置托盘图标(“d:\book. ico”,“* * * 程序”)

上述程序代码表示:在 D 盘的根目录下有“d:\book. ico”图标文件,并且作为图标显示在系统托盘中。提示行为“记事本工具”。

以上分别以 3 种方式指定图标数据(图标资源、图标字节集数据、图标文件名),第 1 种方法更常用。

置托盘图标(,)

上述代码表示清空托盘中的图标。

5. “弹出托盘菜单()”方法

功能:弹出托盘菜单。

语法:窗口名称. 弹出托盘菜单(欲弹出的菜单)

应用实例:

弹出托盘菜单(小菜单)

弹出一个名为“小菜单”的菜单,此菜单由菜单编辑器生成。

在实际应用中,一般先在窗口“托盘事件”的处理程序中判断用户的操作类型(左键单击、左键双击,右键单击)。如果经判断用户单击了右键(或单击了左键),则用“弹出托盘菜单()”命令弹出托盘菜单(如果经判断用户是双击了左键,一般要显示程序主窗口)。

例如软件“网络蚂蚁”的托盘图标,鼠标双击托盘图标时显示程序主窗口,鼠标右击托盘图标弹出托盘菜单,鼠标左击则没有反应。

新建一个易程序,展开程序面板中的“资源表”,单击“图片或图片组”项,在名称下方输入一个名字,在内容列中单击一下,就会弹出一个对话框,如图 5-37 所示。

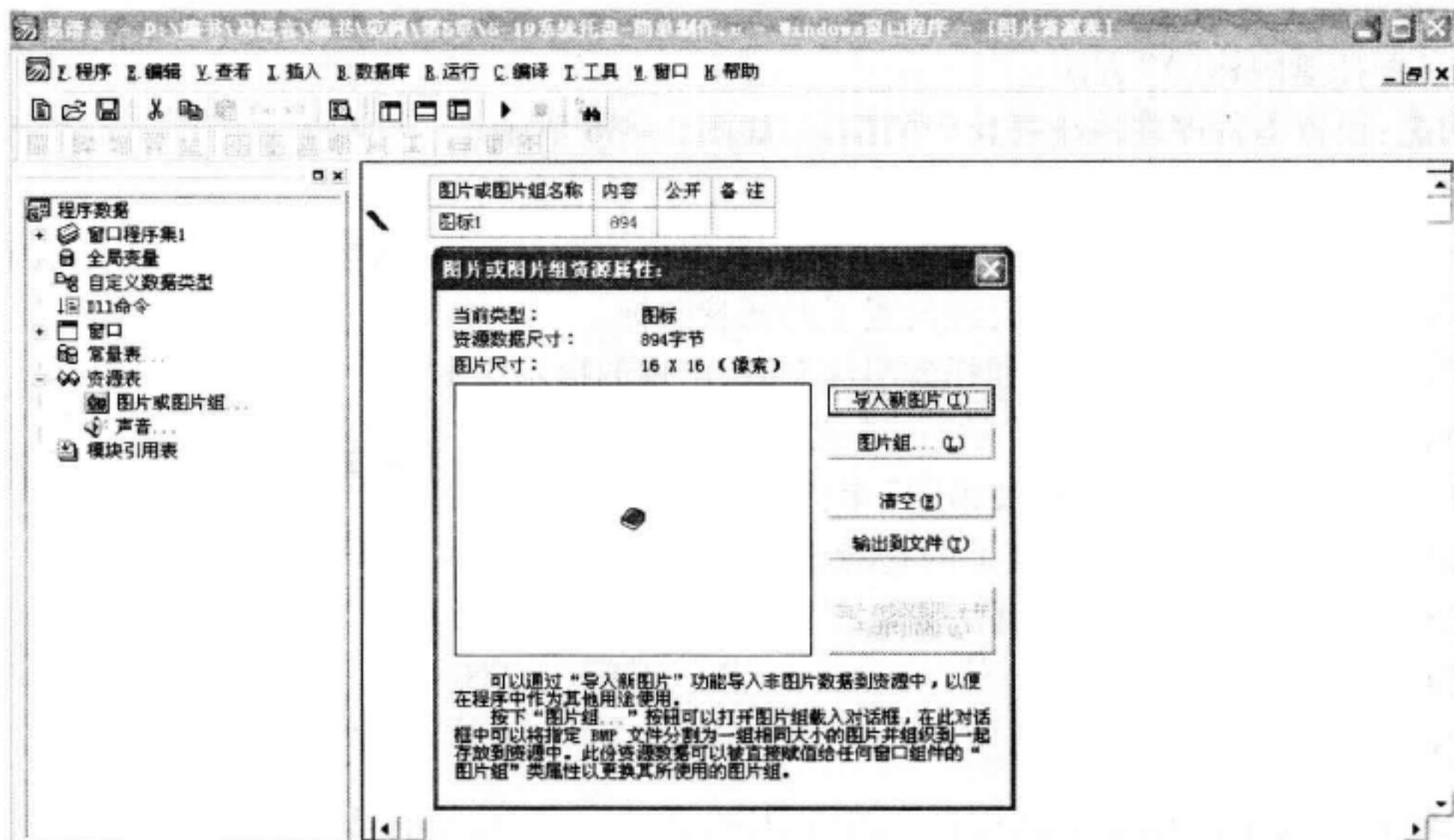


图 5-37 加入一个图标到图片资源中

也可以使用主菜单命令“插入”→“资源”→“图片或图片组”,如图 5-38 所示。

加入小图标完成后,就会看到“内容”中有字节的大小,这表明已加入图标文件。

展开程序面板,双击其中的“_启动窗口”回到启动窗口后,双击窗口中的任意位置,可以进入“__启动窗口_创建完毕”的子程序设计界面如图 5-39 所示,输入程序:

置托盘图标(#图标 1,“记事本工具”)

试运行一下,可以看到在桌面右下角的系统托盘区就出现了一个书样的小图标。具体参见例程 5-22。

另外,还可以用这个控制命令实现以下功能:

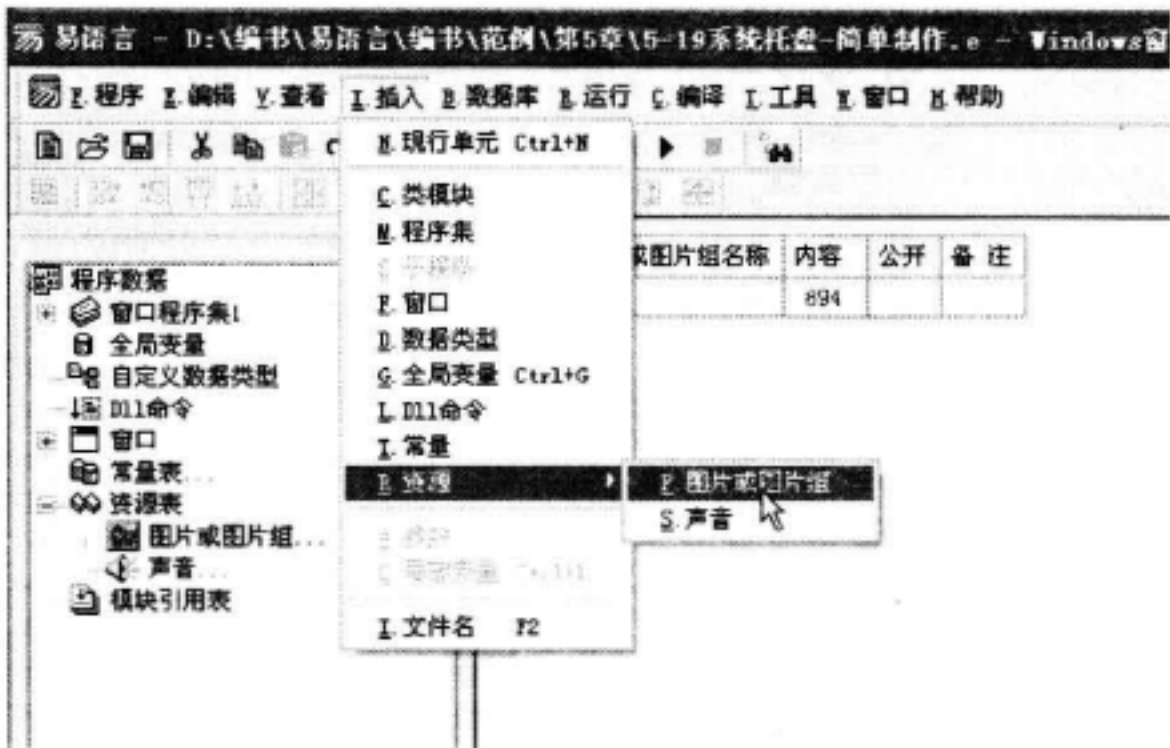


图 5-38 使用菜单命令加入图片资源

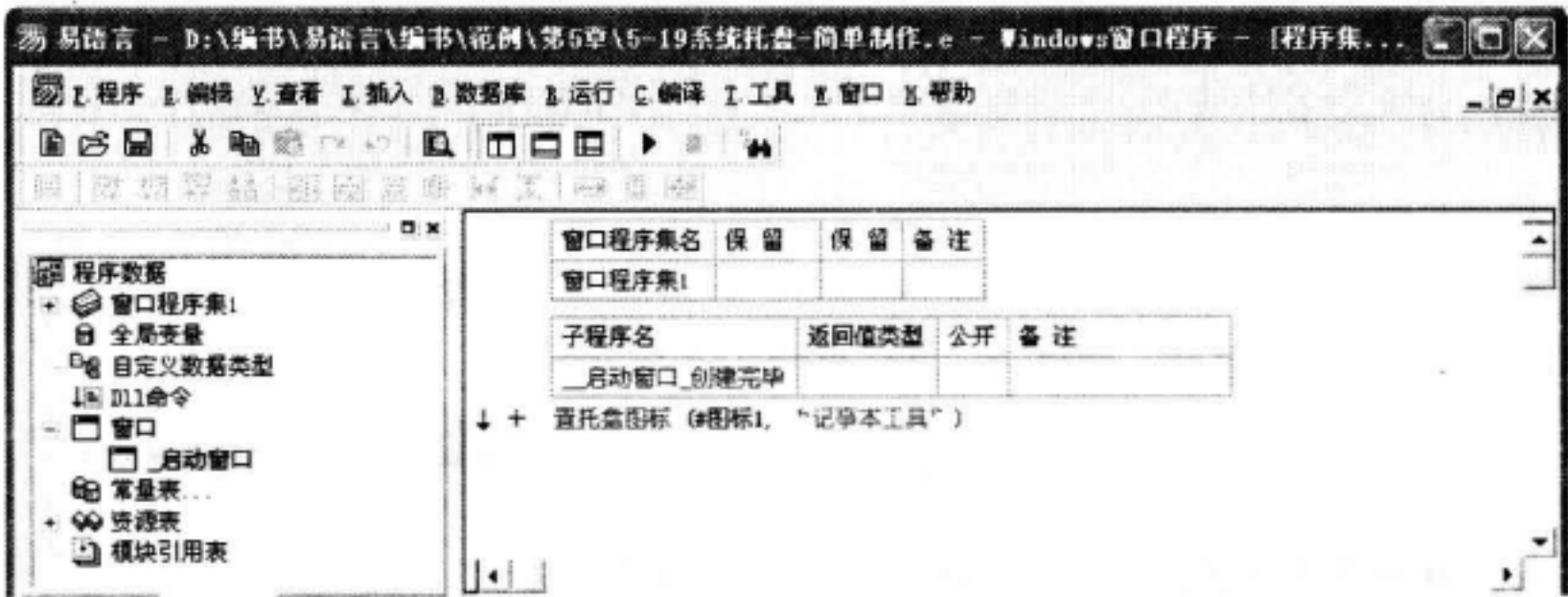


图 5-39 输入程序代码

(1) 启动窗口启动时显示在屏幕上,同时在系统托盘中生成图标,并且有弹出菜单。
具体参见例程 5-23。首先,激活启动窗口后,选属性面板中的事件下拉菜单,选中“创建完毕”选项,进入“__启动窗口_创建完毕”子程序设计界面,输入以下代码:

置托盘图标(#图标图片,)

当然,图标图片是在资源中设置好的。

在这里,还涉及到一个自制窗口控制按钮的方法。将主窗口属性面板中的“控制按钮”属性设置为“假”,而使用 3 个按钮组件进行代替,这样有利于后面的编程,如图 5-40 所示。

最小化替代按钮输入的程序为:

```
子程序:_按钮 1_被单击
_启动窗口.位置 = 2
_启动窗口.可视 = 假
置托盘图标(#图标图片,)
```

最大化替代按钮输入的程序代码为:

```
子程序:_按钮 2_被单击
_启动窗口.可视 = 真
_启动窗口.位置 = 3
```

关闭替代按钮输入的程序代码为:

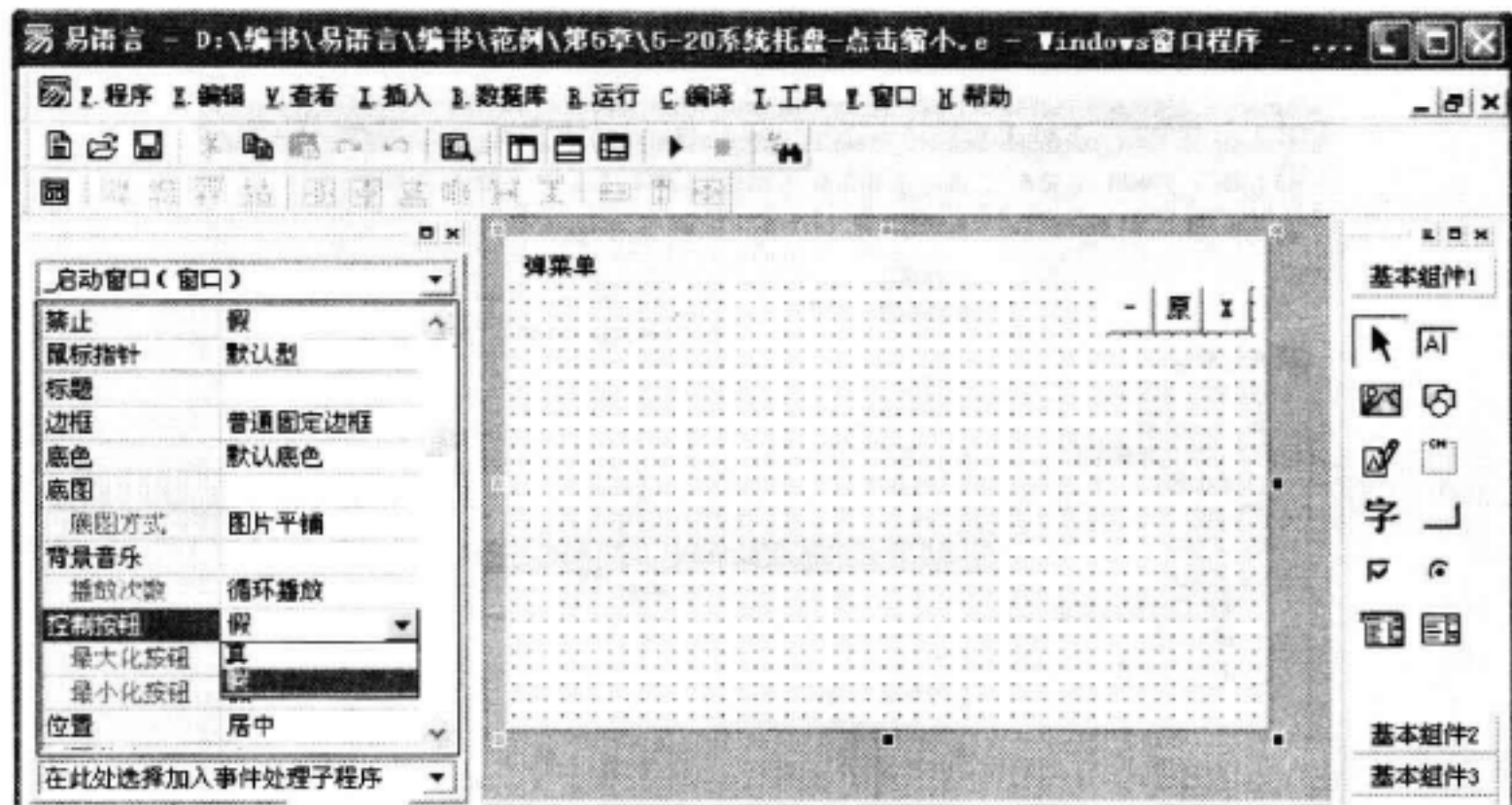


图 5-40 将启动窗口的控制按钮属性设为“假”

```
子程序:按钮3_被单击  
置托盘图标(,)  
结束()
```

本程序中还使用了一个菜单,用作为鼠标击系统托盘时弹出菜单之用。在窗口中使用鼠标单击右键可以弹出一个菜单,如图 5-41、图 5-42 所示。



图 5-41 调出菜单编辑器

用鼠标单击启动窗口,并且选属性面板下面的事件下拉菜单,选中其中的“托盘事件”选项,进入“__启动窗口_托盘事件”子程序编辑,输入如下代码,如图 5-43 所示。

以上内容演示了系统托盘的基本操作,在弹出的菜单中也可以加入程序代码,用来现还原窗口的功能。退出功能也可以加在弹出菜单中。

用鼠标单击标题为“还原”的子菜单,选中“还原”子菜单松手后,即会进入“_还原_被选择”的子程序设计界面,如图 5-44 所示。

输入以下程序代码:

```
子程序:_还原_被选择  
_启动窗口.可视 = 真  
_启动窗口.位置 = 0  
置托盘图标(,)
```

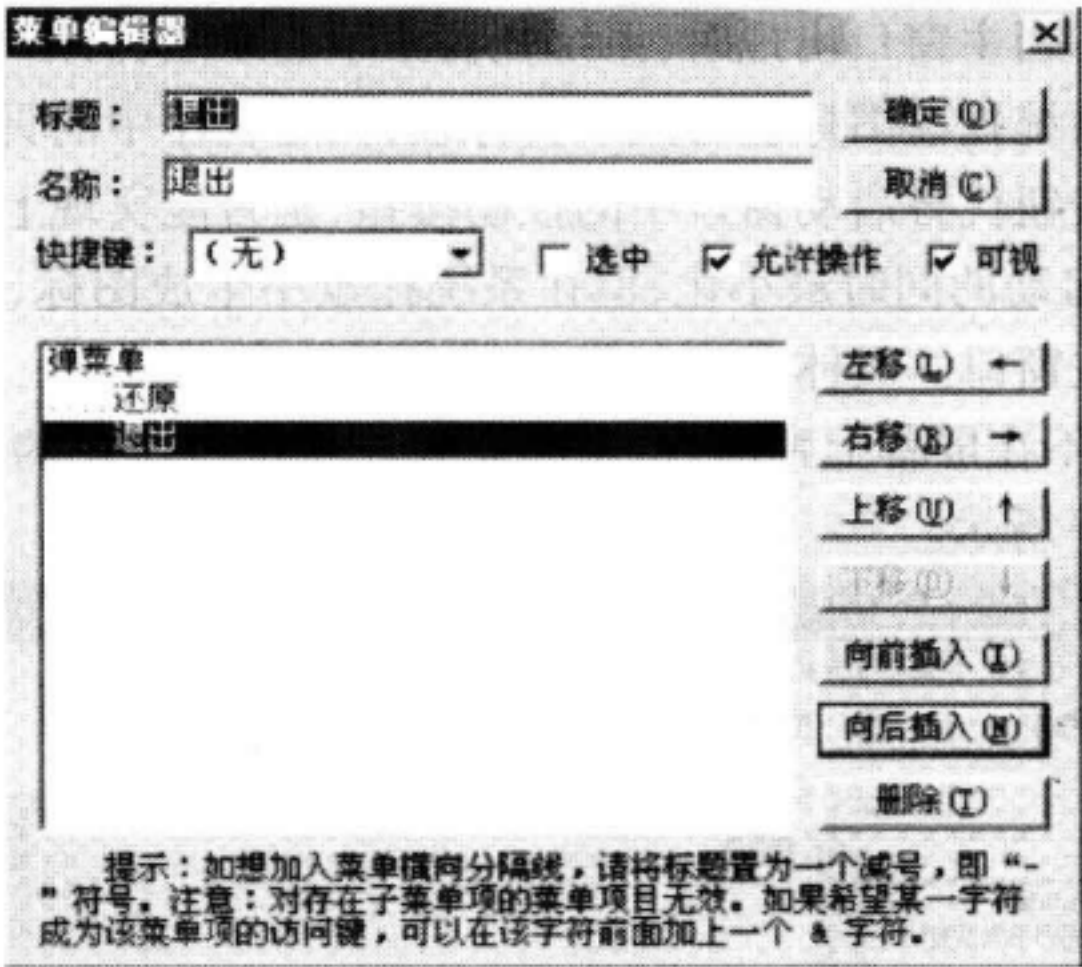



图 5-42 编辑菜单

子程序名	返回值类型	公开	备 注		
__启动窗口_托盘事件					
参数名	类 型	参考	可空	数组	备 注
操作类型	整数型				

```
判断 (操作类型 = #双击)
    _启动窗口.可视 = 真
    _启动窗口.位置 = 0
    置托盘图标 ( )

判断 (操作类型 = #单击右键)
    弹出托盘菜单 (弹菜单)
```

图 5-43 输入程序代码

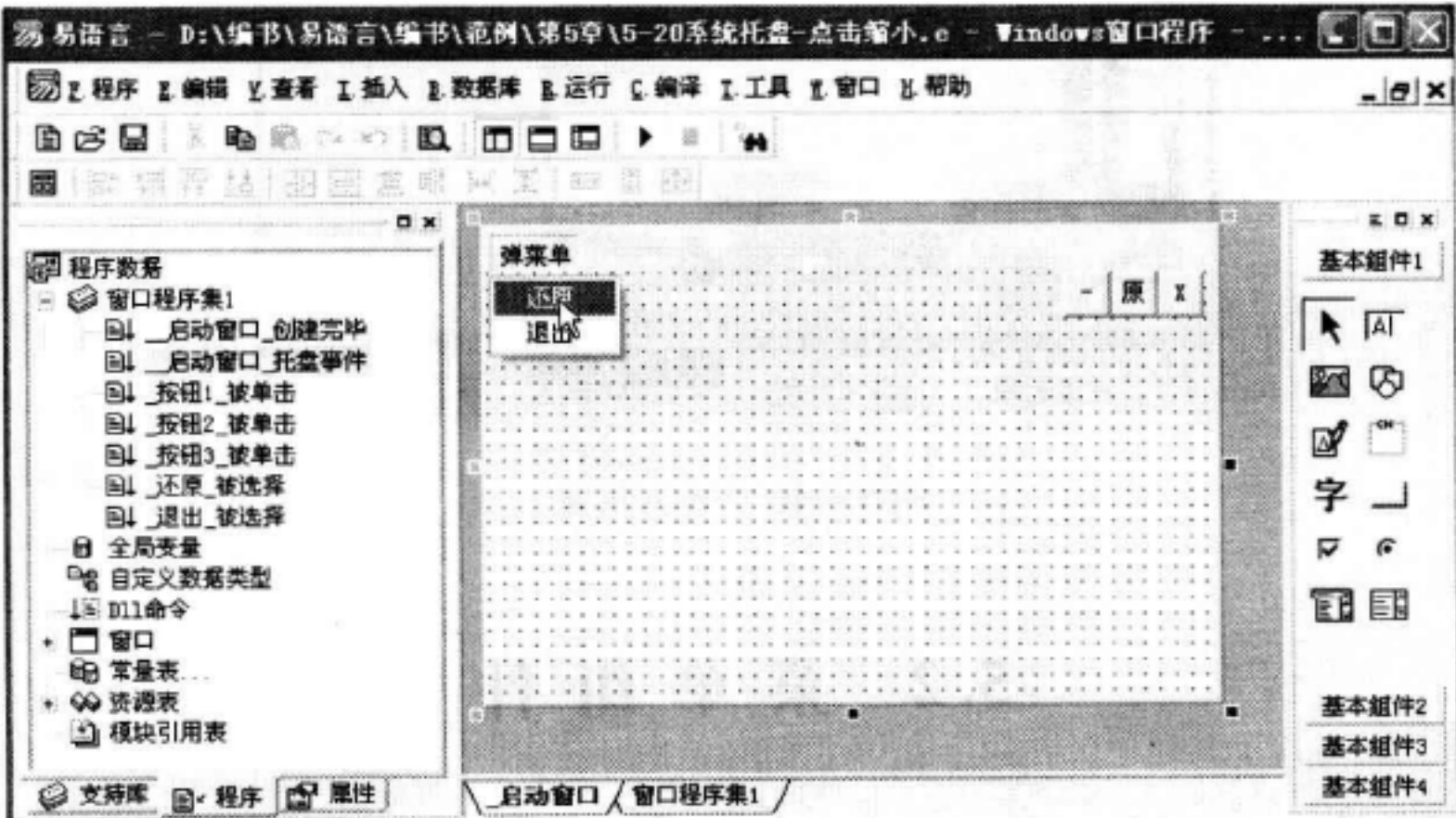


图 5-44 选择要编写事件代码的菜单项

本例还可以设计为:当主窗口出现时,系统托盘中不显示图标;当主窗口最小化时,系统托盘才显示图标。当然,系统托盘图标也可以一直存在,只看程序中的具体要求了。

试一试:如果不用按钮代替启动窗口中的控制按钮,是否能实现上述功能。

(2) 主启动窗口在启动的同时最小化,但在系统托盘中生成图标,并且鼠标单击时会弹出菜单,由菜单显示弹出主窗口。具体参见例程 5-24。

启动窗口启动时即不在屏幕上弹出,这个功能只要直接将窗口的属性面板中可视属性改为“假”即可,如图 5-45 所示。



图 5-45 改启动窗口可视属性为“假”

(3) 图标在程序运行中一直在变化,形成滚动的效果。弹出菜单可以控制播放或停止,同时系统托盘中的图标也变为相应的状态,如图 5-46 所示。

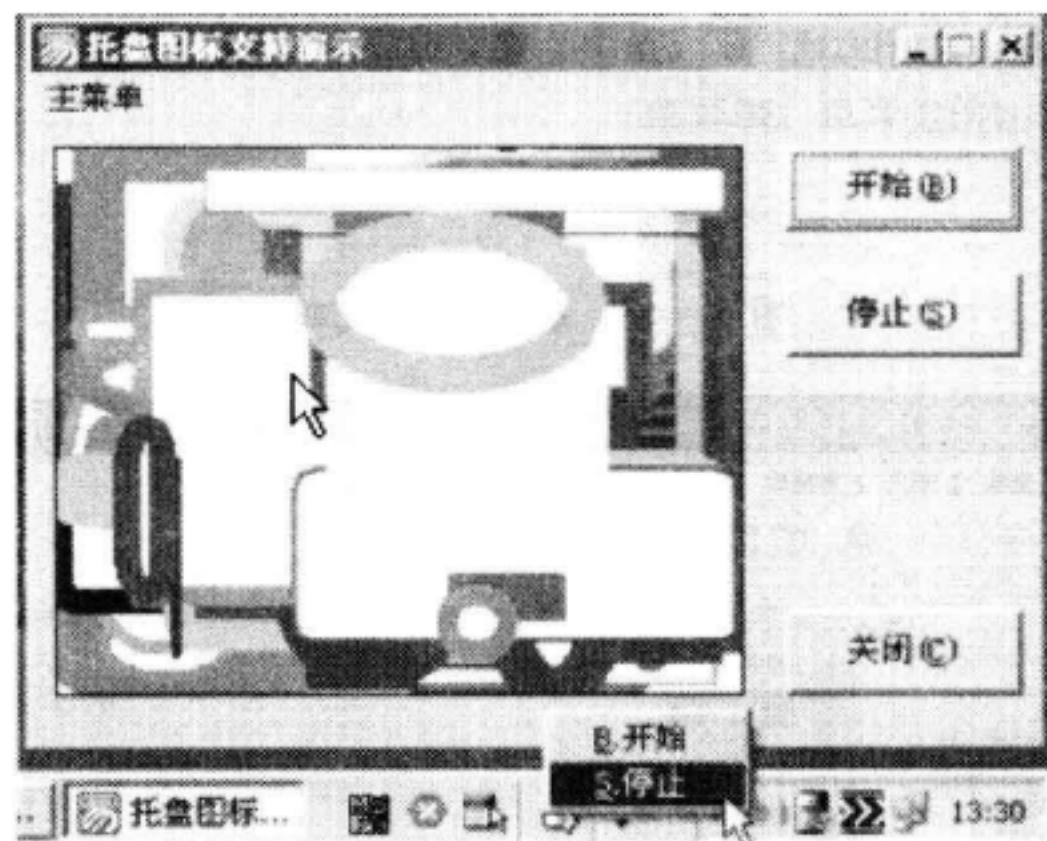


图 5-46 运行结果

5.2 菜单组件

菜单作为 Windows 程序特别是早期 Windows 程序的最突出的特点,早已成为用户友好界面必不可少的组成部分。它不仅使用户操作变得简单和方便,而且也使程序的界面更加美观。过去,要为一个程序开发一组菜单并不是一件容易的事情。而现在,通过易语言强大的集成开

在单击“确定”按钮后,进入设计界面可以发现菜单已经建立好了。单击主菜单会拉出下级菜单,与实际运行效果一样,如图 5-50 所示。



图 5-50 设计时的子菜单

而对于二级子菜单、三级子菜单来说,只要多单击几次“右移→”按钮,就可以同样的方法实现。

如何对菜单内进行事件编程呢?选中某一个子菜单项后,会自动进入程序代码设计界面。例如:用鼠标单击“帮助”→“关于”菜单,会产生“_关于_被选择”的子程序,如图 5-51 所示。程序代码设计如图 5-52 所示。

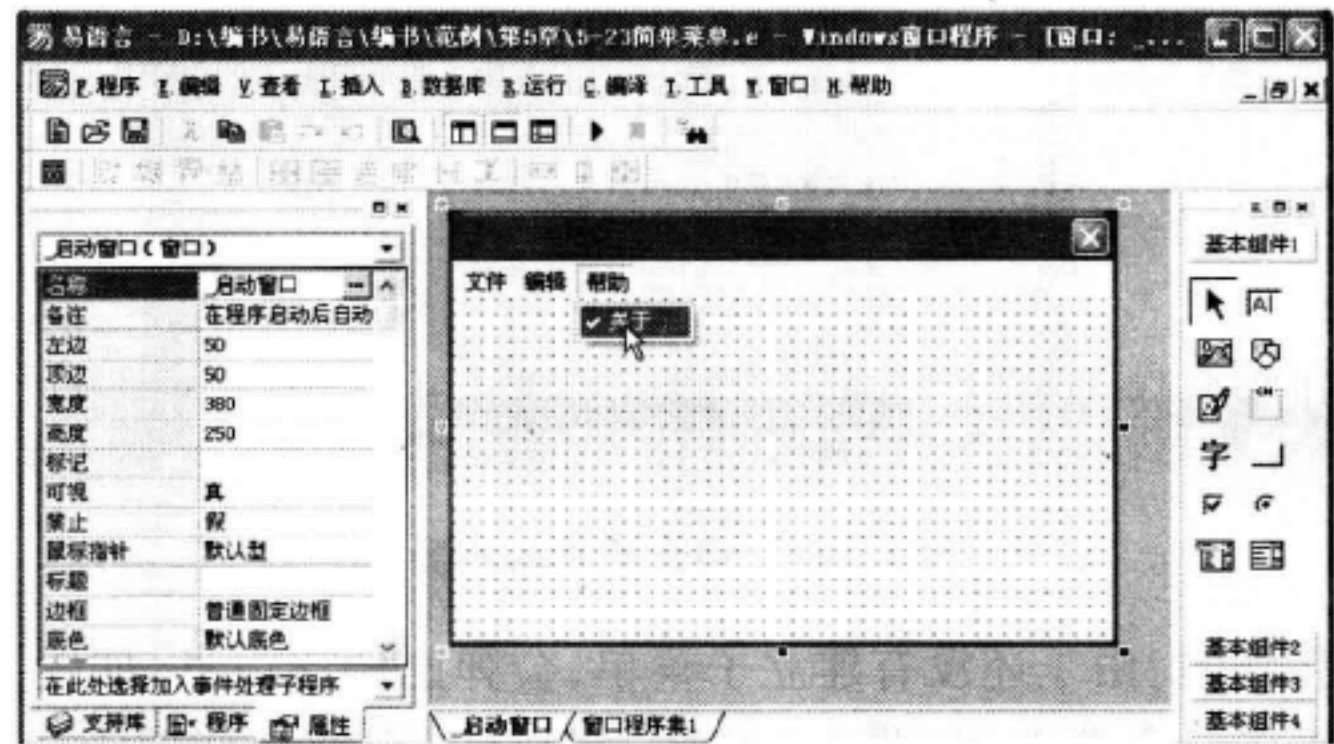


图 5-51 菜单事件

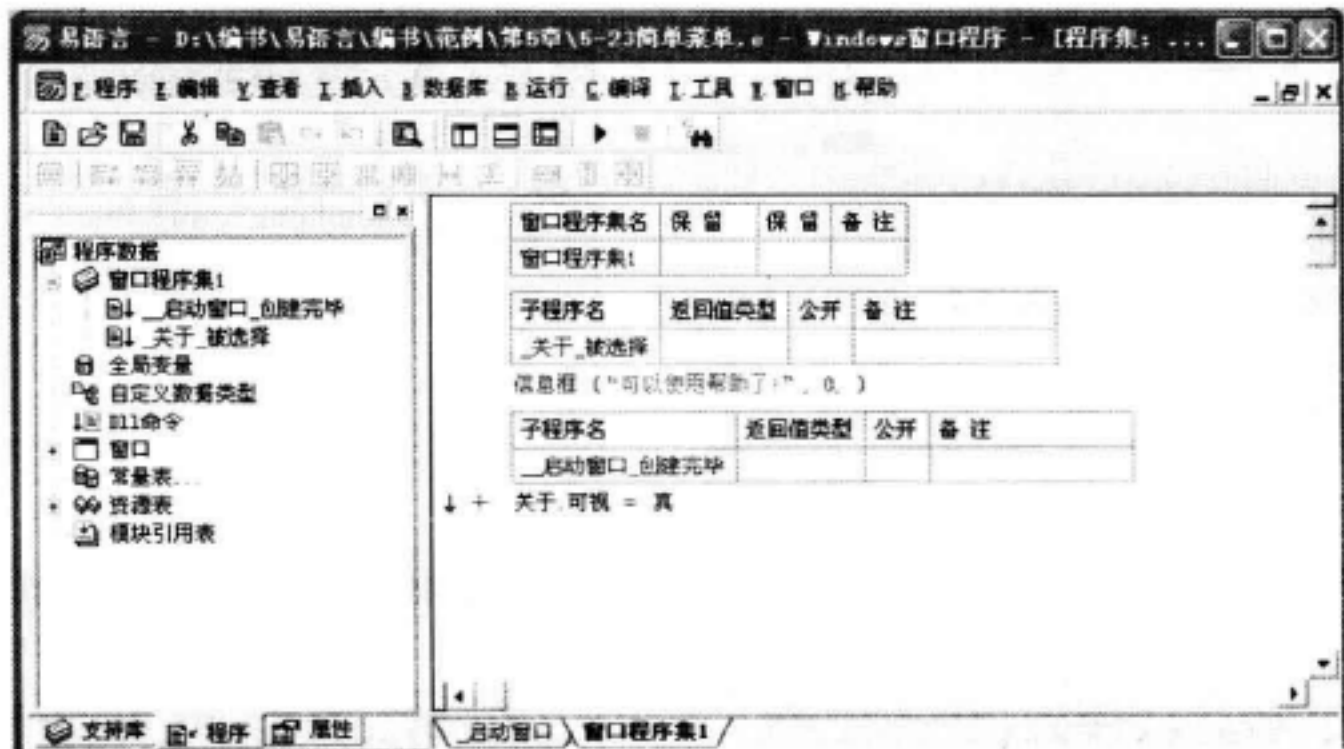


图 5-52 程序代码设计

这表示当菜单“关于”被按下后,会进入这个“_关于_被选择”子程序,并执行这个子程序内的命令。输入下面一行代码,然后试运行,看看运行后的结果。

子程序:_关于_被选择

信息框(“可以使用帮助了!”,0,)

菜单还有其他一些特性,如“选中”、“允许操作”、“可视”。

“选中”选项可以控制是否在子菜单前面加勾。

“允许操作”选项可以控制子菜单是否可以操作。如果控制当前菜单不允许操作,则在运行时该菜单会以灰色显示,表示不可用。

“可视”选项可以控制子菜单是否可以看见。

【注意】其中有一个子菜单项是一个减号“-”,它表示分栏线。

选择不同的复选框试运行,会有不同的效果,如图 5-53 所示。

操作这些选项的方法也非常简单。只要在程序中输入“菜单名称. 选项 = 真”或“菜单名称. 选项 = 假”即可。如果要将关于子菜单的可视属性改为非选中状态,输入以下代码即可。

关于. 可视 = 假

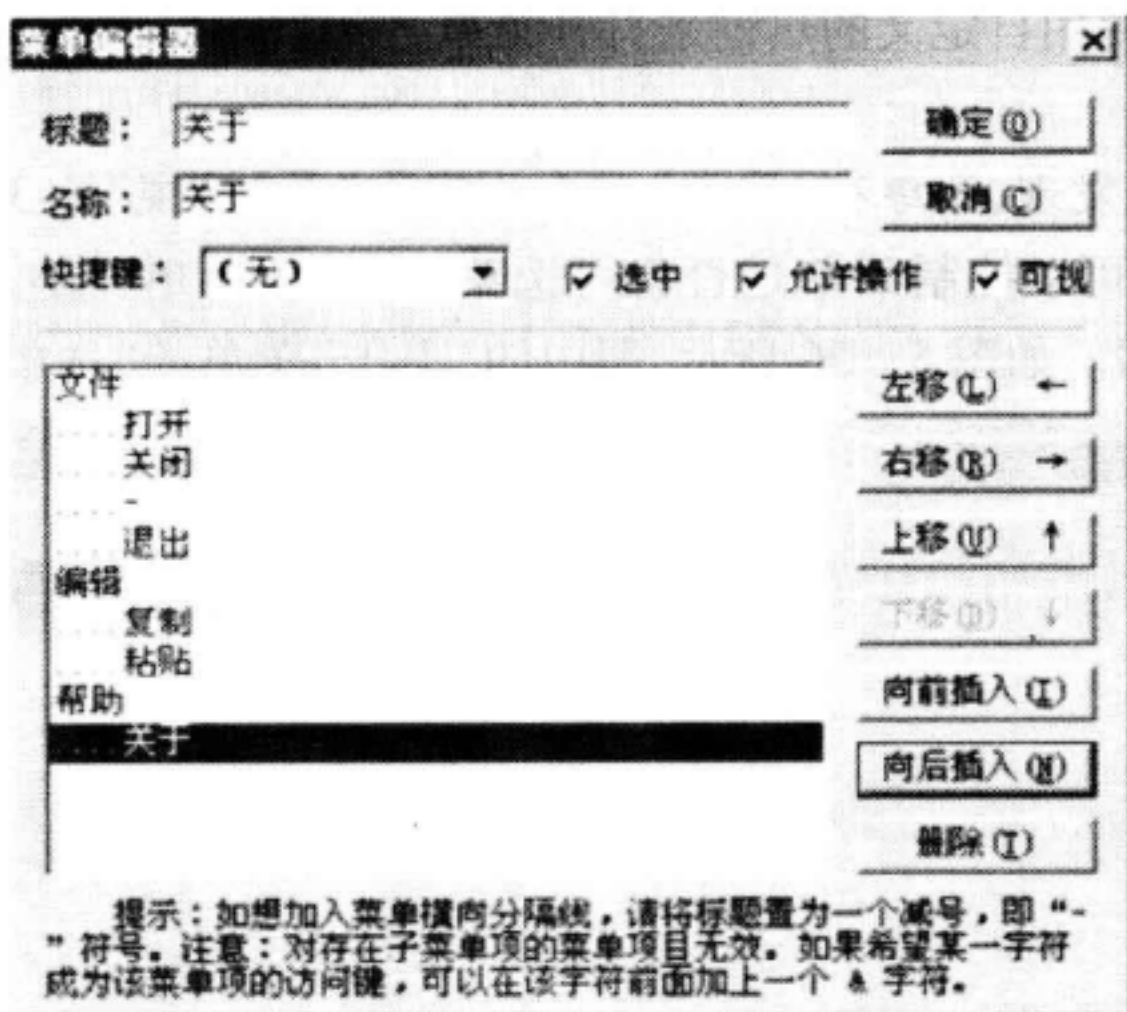


图 5-53 菜单中的选项

【注意】其他的选项的控制方法相同。

如图 5-53 所示,菜单编辑器的最上两排分别为“标题”与“名称”。在输入“标题”时,“名称”自动会与“标题”同名,但也要注意以下的输入原则:

“标题”表示显示在菜单上的文字,标题与标题之间可以重复,但“名称”是真正在程序中指定的关键词,因此不能重复,否则系统不知道该引用哪一个菜单,会报错。标题也同样可以更改。具体参见例程 5-25。

5.2.2 菜单的热键与属性

1. 菜单的热键

可以为菜单项设置快捷键,即按住 Alt 键后,同时按下另一个键,就可以调用指定的菜单项功能。一般情况下,有两种表示方法:

菜单名(&F)或 &F 菜单名

例如将图 5 - 53 编辑过的菜单都加上热键,如图 5 - 54 所示。

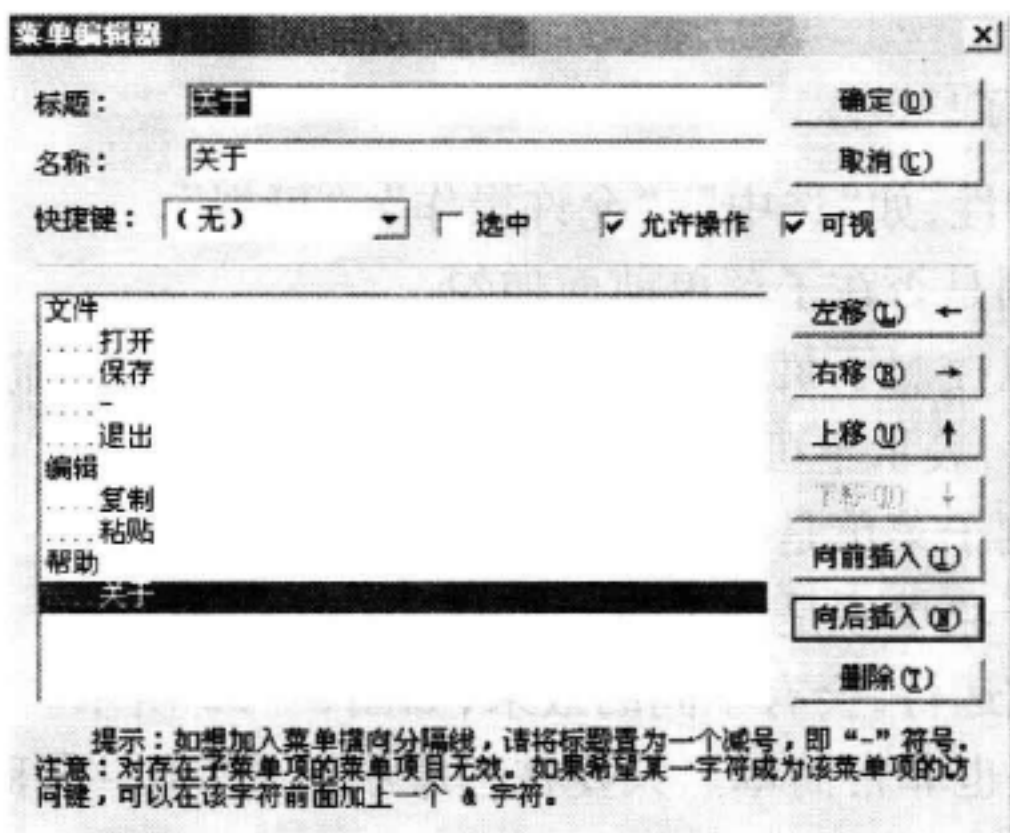


图 5 - 54 菜单加热键

程序运行后,可以使用自定义的热键来打开菜单,如图 5 - 55 所示。

2. 菜单的属性

在编辑菜单项时会发现,菜单有“选中”、“允许操作”及“可视”这 3 个属性。

(1)“选中”属性。可以控制菜单是否为可选菜单,一个菜单“选中”属性被选择后,该菜单的前面就会出现“√”。如图 5 - 56 所示,图中的“关于”菜单的“选中”属性被选择。

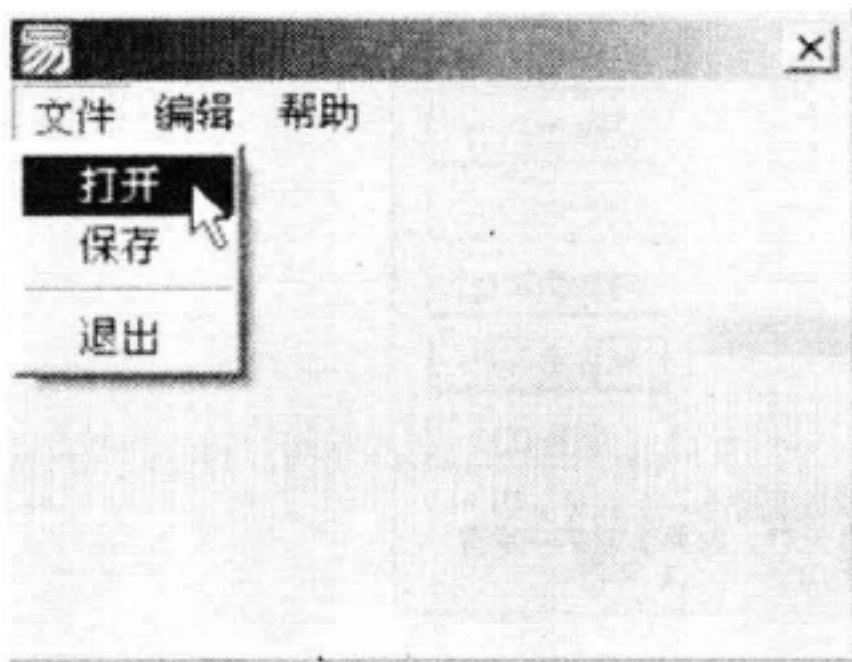


图 5 - 55 用热键打开菜单

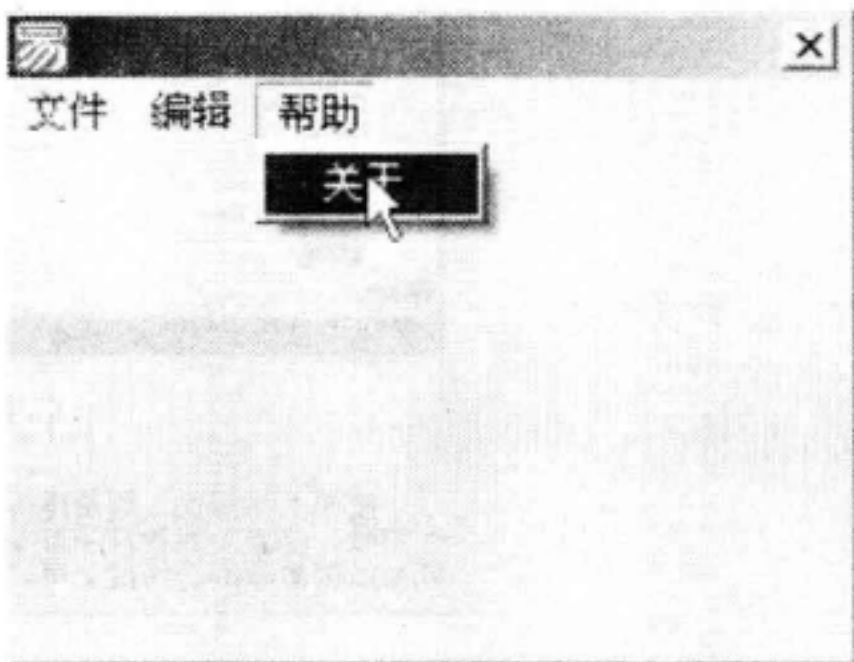


图 5 - 56 菜单的“选中”属性

(2)“允许操作”属性。菜单的“允许操作”属性类似按钮组件的“禁止”属性,当菜单的“允许操作”属性被取消选择后,菜单就变成灰色,将不能对该菜单进行任何操作。如图 5 - 57 所示。图中的“文件”菜单的“允许操作”属性被禁止。

(3)“可视”属性。当菜单的“可视”属性被取消选择后,该菜单项就会隐藏,程序运行过程中不可见。如图 5 - 58 所示。图中的“编辑”菜单被取消“可视”,所以运行后看不到该菜单项。

(4)菜单的被选择事件。要实现菜单上各选项的功能,就需要在该项菜单被选择事件子程序中加入相应的代码。在窗口中点击菜单中的一个选项,就会自动生成该项被选择事件子程序。

例如,图 5 - 53 编辑过的菜单中的“退出”选项,加入结束程序运行的功能。需要在菜单中选择“退出”选项,然后在生成的“_退出_被选择”子程序中输入代码:

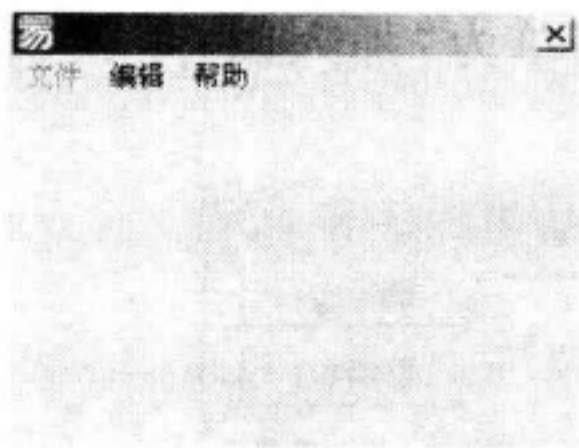


图 5-57 菜单的“允许操作”属性

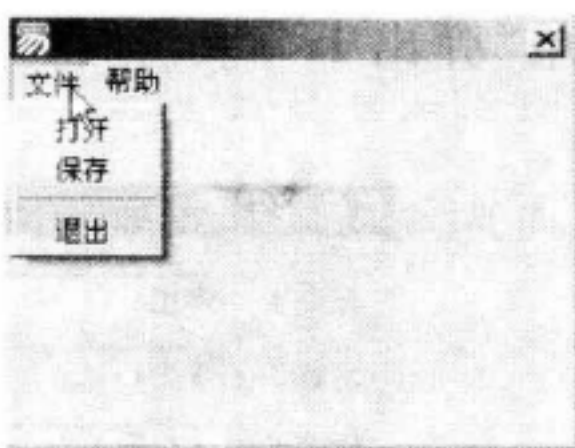


图 5-58 菜单的“可视”属性

结束()

(5)使用代码控制菜单属性。

菜单的属性可以在程序中用代码进行控制,例如:将“编辑”菜单的“可视”属性设置为假:

编辑. 可视 = 假

还可以通过改变菜单的“选中”属性,让该菜单项选中和取消选中,例如,刚才编辑过的菜单,点击菜单中的“关于”选项,在“_关于_被选择”子程序中输入代码:

子程序名	返回值类型	公开	备注
_关于_被选择			
如果 (关于.选中 = 真)			
关于.选中 = 假			
关于.选中 = 真			

当程序运行后,就可以显示和取消“关于”前的“√”。

具体参见例程 5-26。

5.2.3 弹出菜单

单击鼠标右键弹出一个菜单,那么这样的功能如何实现呢?

建立一个菜单条,将其中的主菜单的“显示选项”设置为“不显示”。然后在程序代码中输入:

弹出菜单(弹出菜单名, ,)

新建一个易程序,在其中放置一个画板,画板的边框设置为镜框式,如图 5-59 所示。

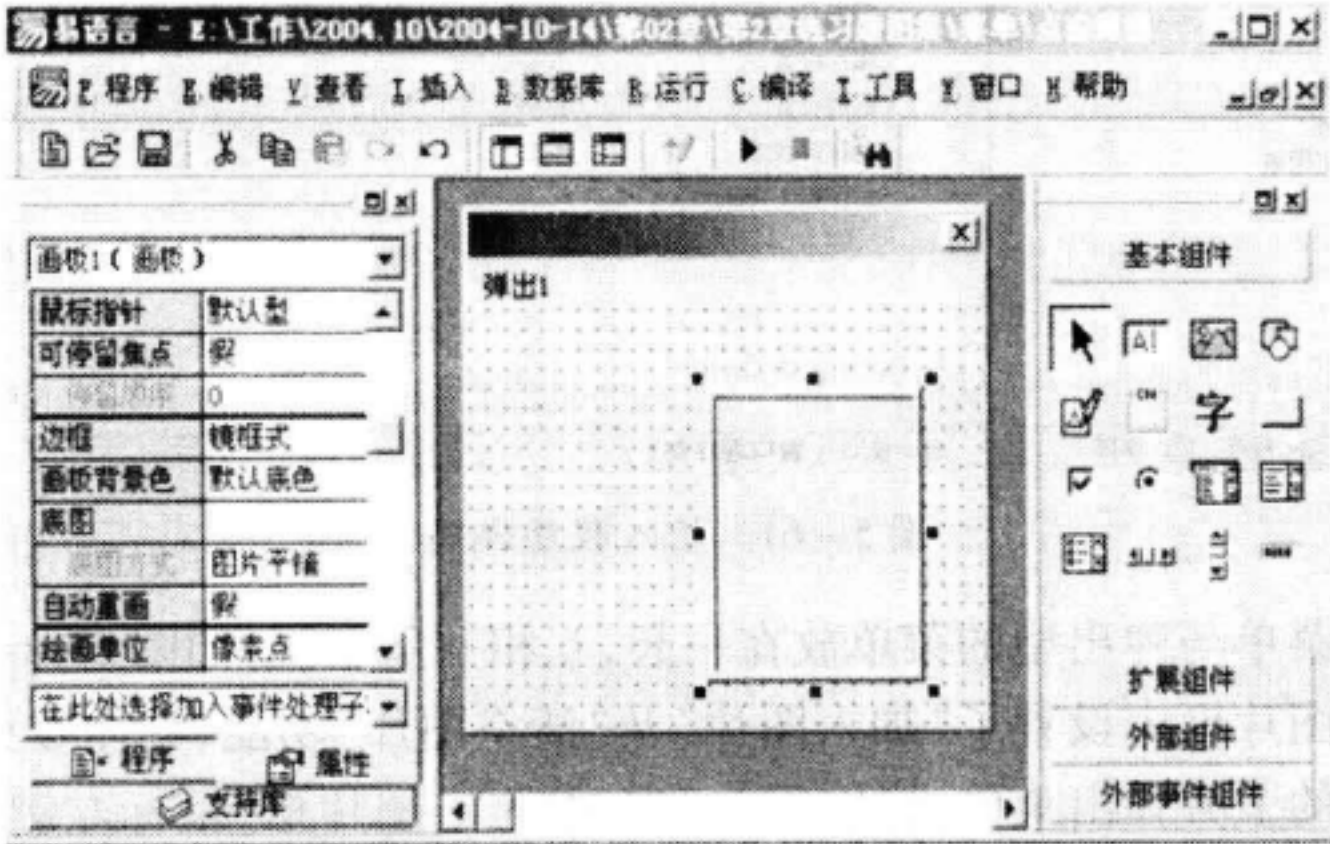


图 5-59 在启动窗口中添加一个图片框组件

建立主菜单“弹出 1”，取消其“可视”选项，子菜单为“加入图片”、“清空图片”，如图 5-60 所示。



图 5-60 设计菜单

激活画板，单击属性面板最下方的事件下拉菜单，找到“鼠标左键被按下”事件，在“_画板 1_鼠标左键被按下”事件的子程序中输入以下代码，如图 5-61 所示。

弹出菜单(弹出 1, ,)

试运行该程序，当鼠标在镜框中单击时就会产生菜单，而窗口界面上没有菜单。具体参见例程 5-27。

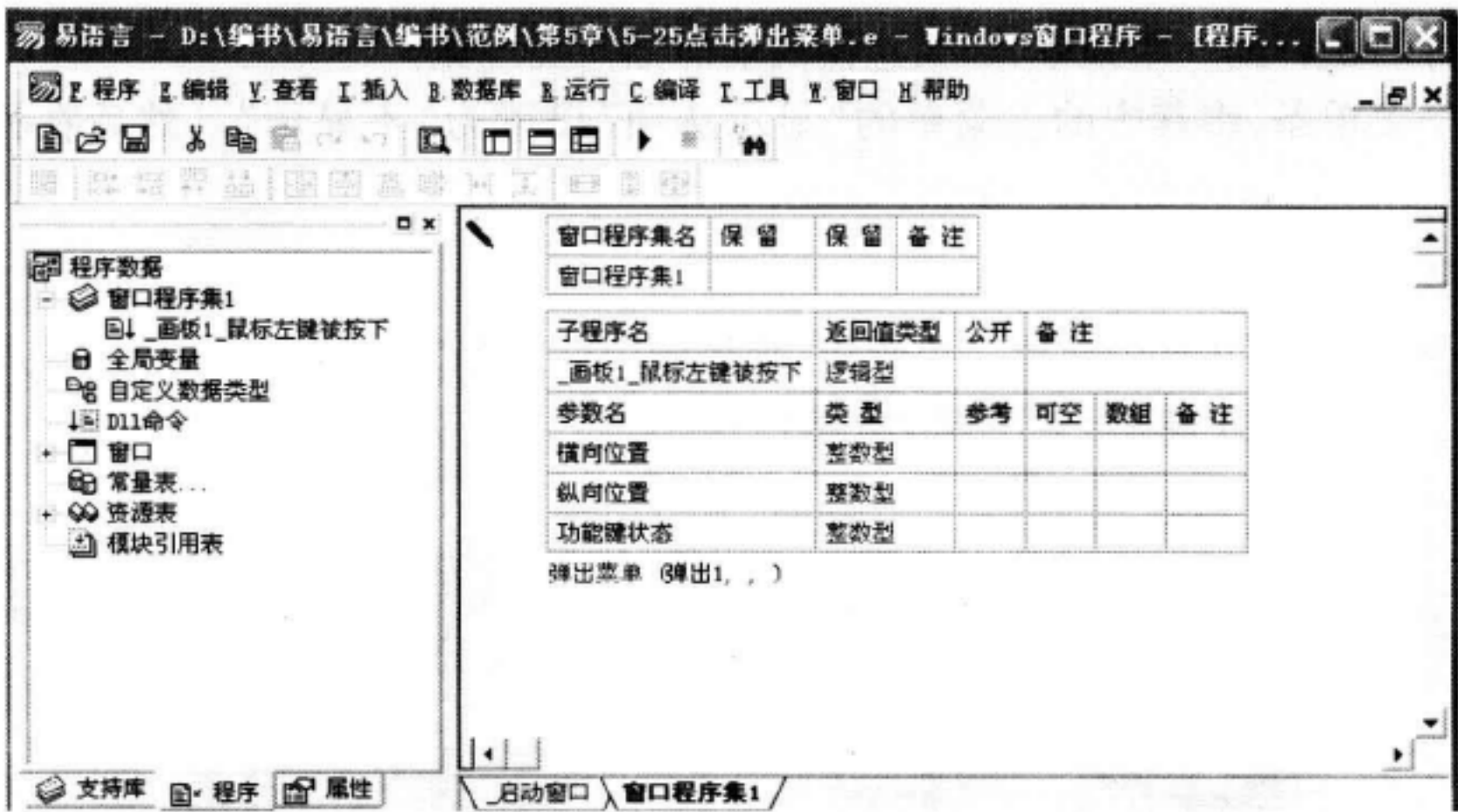


图 5-61 输入程序代码

可以将普通的菜单与弹出型的菜单放在一起，互相并不干扰，如图 5-62 所示。这个例程已为图片框中设置了“加入图片”及“清空图片”功能，具体参见例程 5-28。也可以在鼠标的上方弹出菜单，将例程 5-26 中的“_图片框_鼠标左键被按下”中的原语句“弹出菜单(弹出 1, ,)”改为以下的语句：

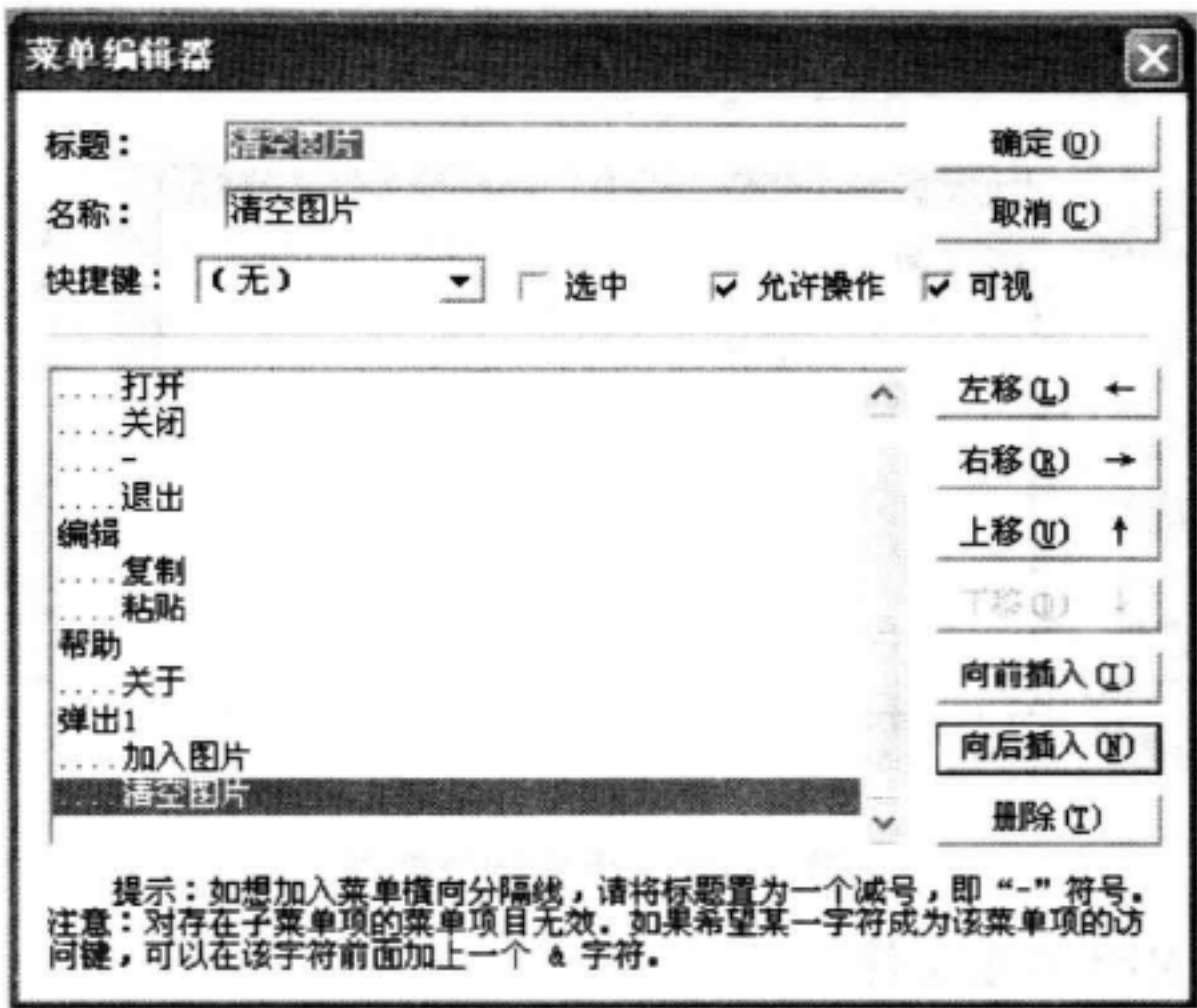


图 5-62 设置普通菜单与弹出菜单

弹出菜单(弹出 1,取鼠标水平位置()-50,取鼠标垂直位置()-80)

再次试运行时即可以看到弹出的菜单在鼠标的上方了,如图 5-63 所示。

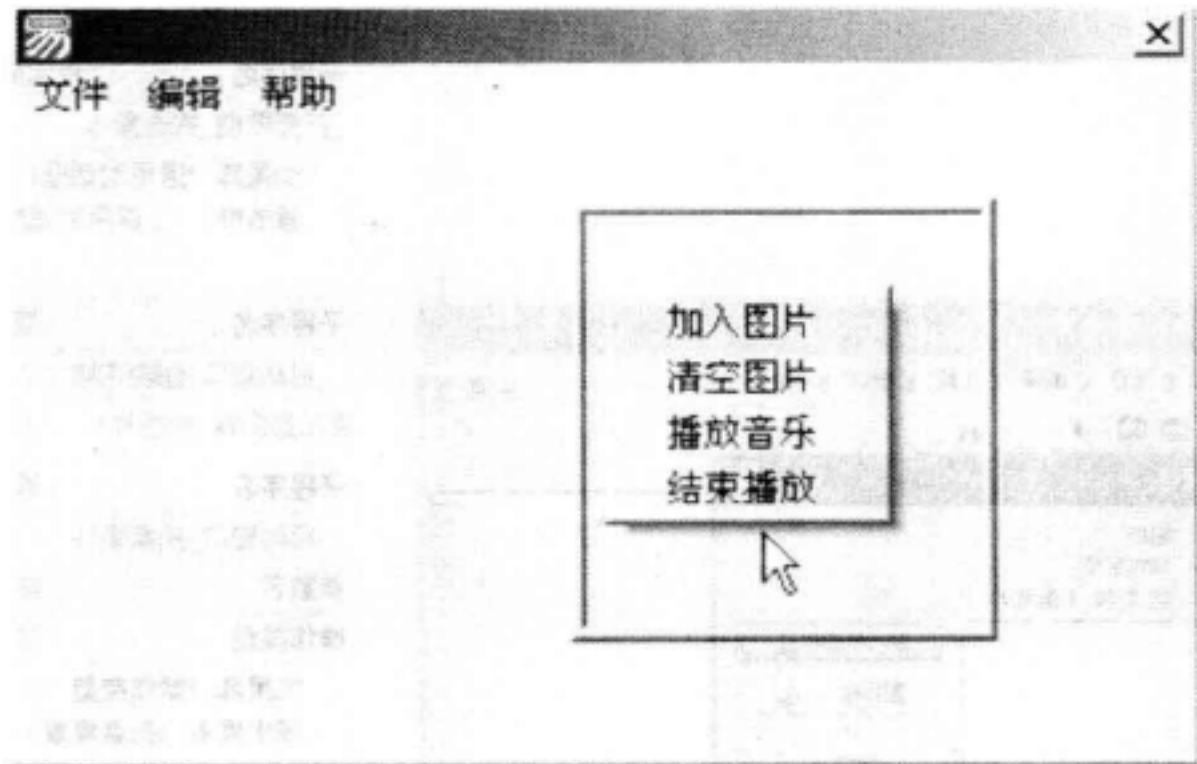


图 5-63 运行结果

原来,这是使用了弹出菜单这个命令中的另两个参数实现的。其中,第 2 个参数是控制菜单的水平显示位置,第 3 个参数是控制垂直显示位置。

对象.弹出菜单(欲弹出的菜单,[水平显示位置],[垂直显示位置])

具体参见例程 5-29。

5.2.4 托盘菜单

新建一个易程序,在启动窗口中设置“可视”属性为“假”。并且制作一个菜单,主菜单为“托盘菜单”。子菜单分别为“打开歌曲”、“停止播放”、“退出”,如图 5-64 所示。

在图片资源中读入一个图标,名称为“图片 1”,如图 5-65 所示。

在“__启动窗口_创建完毕”事件子程序中,完成载入系统托盘图标的动作。

在“__启动窗口_托盘事件”事件子程序中,完成弹出菜单的动作。

在“_打开歌曲_被选择”事件子程序中,完成打开歌曲的动作。打开时利用了一个对话框

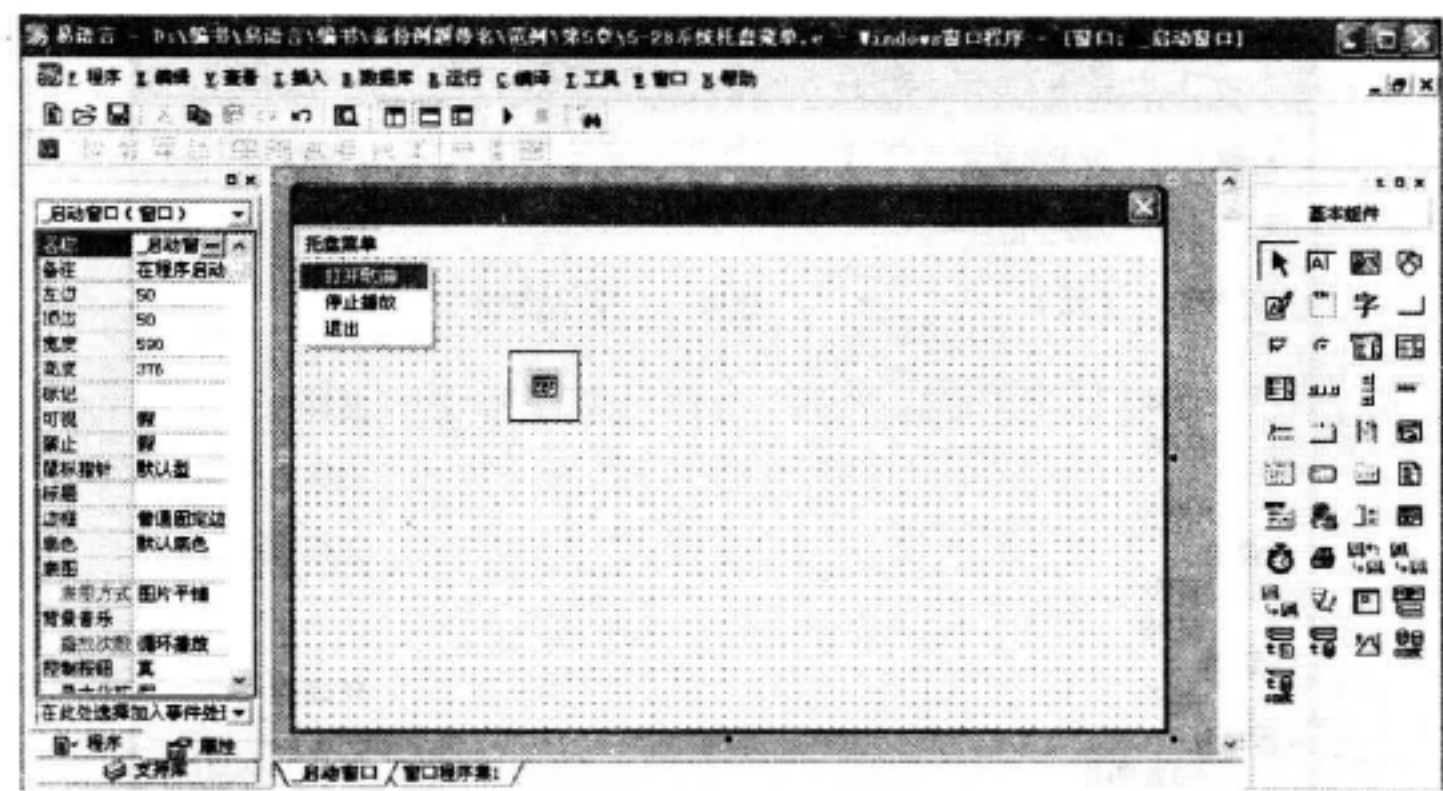


图 5-64 设计托盘菜单

组件来定位要打开的 MP3 音乐文件。

在“_停止播放_被选择”事件子程序中，完成结束歌曲播放的动作。

在“_退出_被选择”菜单事件子程序中，完成系统托盘图标清空及结束程序的动作。相关程序如图 5-66 所示。

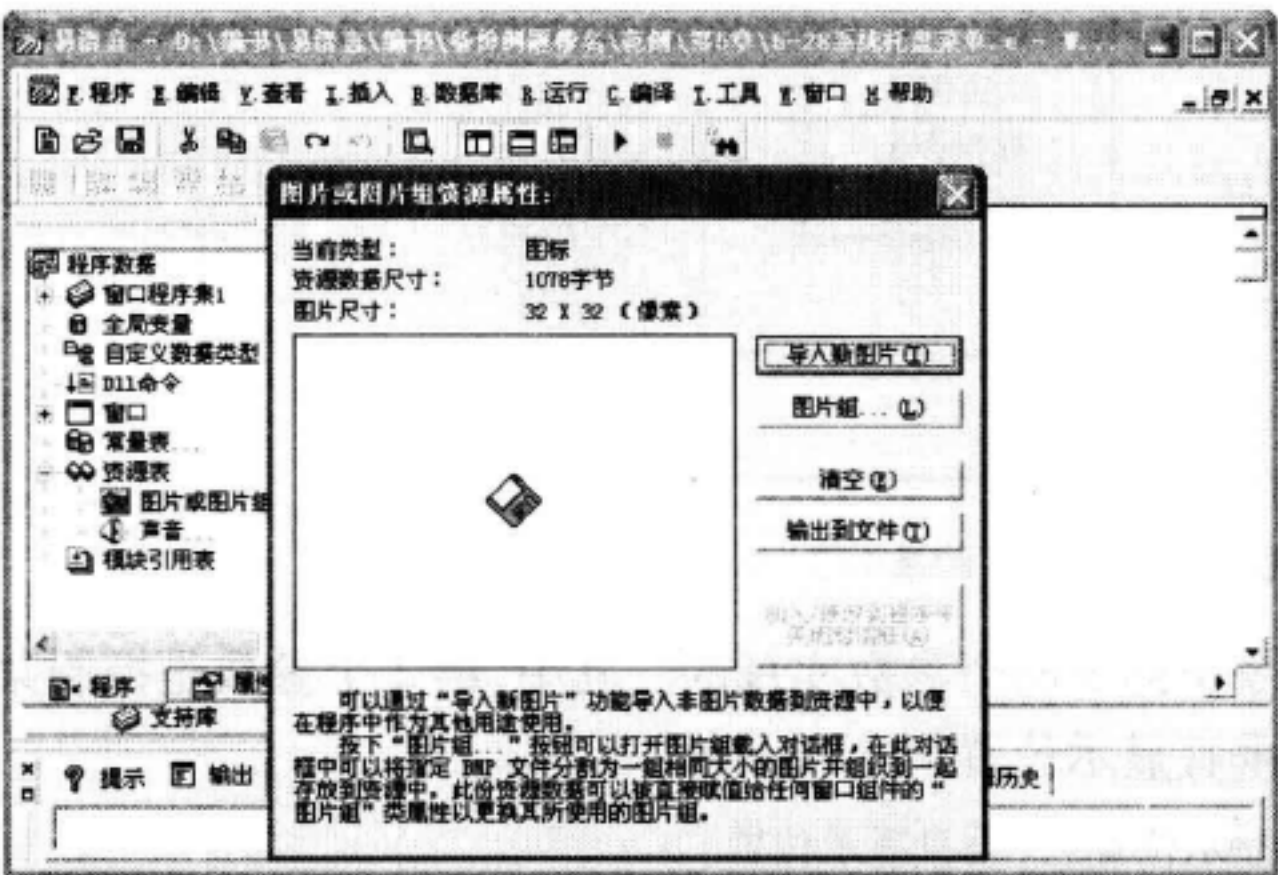


图 5-65 导入图标到图片资源中

窗口程序集名	保留	保留	备注
窗口程序集1			

子程序名	返回值类型	公开	备注
_打开歌曲_被选择			

如果真 (通用对话框1.打开 () = 真)
播放MP3 (通用对话框1.文件名)

子程序名	返回值类型	公开	备注
_启动窗口_创建完毕			

置托盘图标 (#图片1, “托盘图标”)

子程序名	返回值类型	公开	备注
_启动窗口_托盘事件			

参数名	类型	参考	可空	数组	备注
操作类型	整数型				

如果真 (操作类型 = #单击右键)
弹出菜单 (托盘菜单,)

子程序名	返回值类型	公开	备注
_退出_被选择			

置托盘图标 (“托盘图标”)
结束 ()

子程序名	返回值类型	公开	备注
_停止播放_被选择			

停止播放 ()

图 5-66 程序代码

具体参见例程 5-30。

5.3 超级菜单

5.2 节介绍了创建普通菜单的方法，可能会显得太普通了，不能和现在多种多样的图形界面搭配，下面介绍的“超级菜单”组件就可以满足对菜单的各种需要，该组件不但可以设置菜单各部分的颜色，还具有设置菜单标题前图标和接收菜单被点燃的事件等功能。

“超级菜单”组件的基本使用方法很简单,只要程序中添加了该组件,当程序运行后,就会自动将窗口菜单的外观改变成“超级菜单”组件设置后的外观。下面就来了解“超级菜单”组件的具体使用方法。

5.3.1 超级菜单的属性

1. “配色方案”属性

“超级菜单”组件提供了6种供选择的整体菜单的配色方案:(1)蓝色,(2)灰色,(3)雅绿,(4)粉红,(5)青春,(6)古朴。这些配色方案将自动设置以下属性:“背景颜色”、“文本颜色”、“禁止文本颜色”、“点燃颜色”、“渐变条颜色1”、“渐变条颜色2”、“菜单条点燃颜色1”、“菜单条点燃颜色2”、“点燃区边框颜色”、“分割条颜色”。蓝色风格的菜单如图5-67所示。

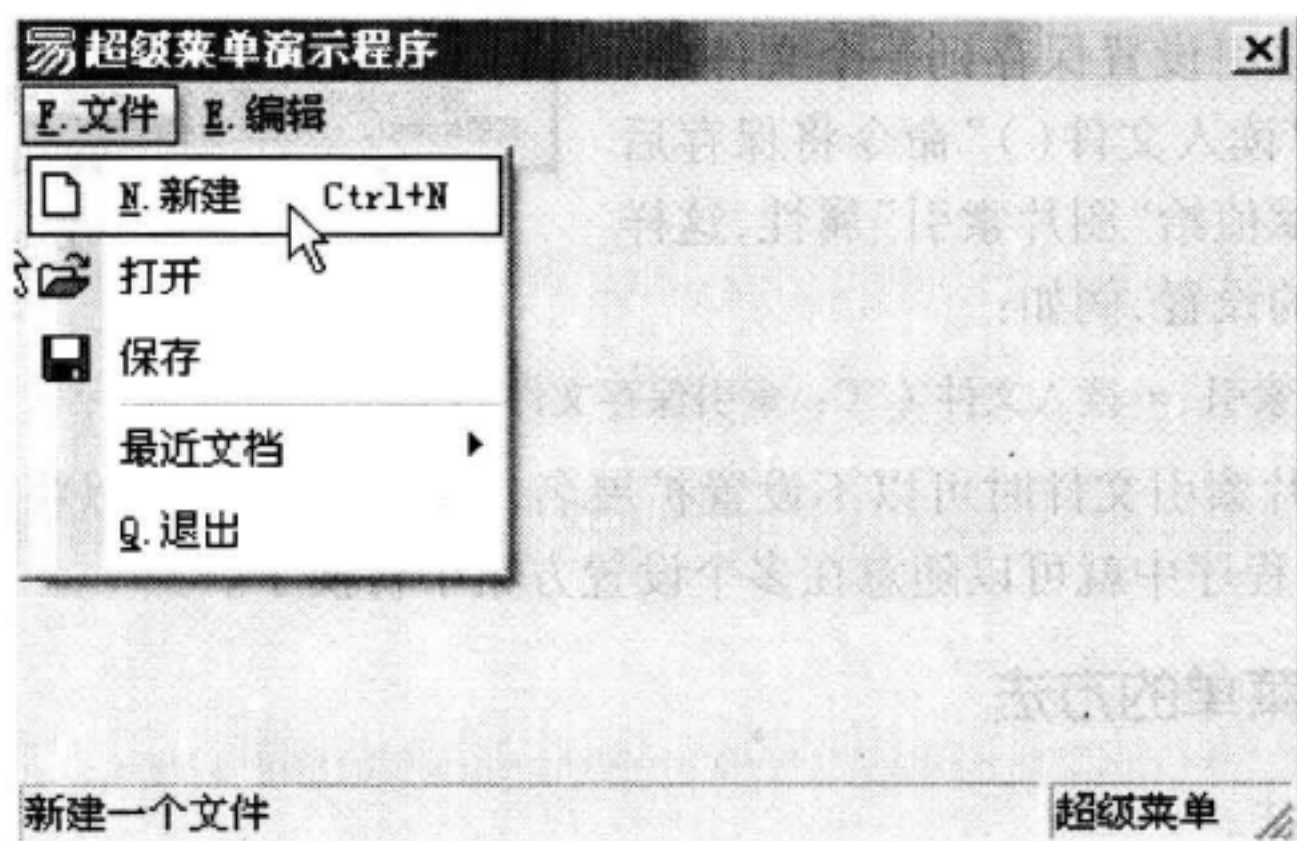


图 5-67 蓝色风格超级菜单

“配色方案”属性也可以设置为0. 自定义。当选择“自定义”后,上述颜色属性就可以自己配色了。

2. “背景颜色”、“文本颜色”、“禁止文本颜色”、“点燃颜色”、“渐变条颜色1”、“渐变条颜色2”、“菜单条点燃颜色1”、“菜单条点燃颜色2”、“点燃区边框颜色”、“分割条颜色”属性。

“背景颜色”属性设置菜单的总体背景,左边的渐变条除外。

“文本颜色”属性设置菜单中的文本颜色。

“禁止文本颜色”属性设置当菜单中的选项被禁止后的文本颜色。

“点燃颜色”菜单中各项目当鼠标停留在其上方时的颜色,但顶级菜单除外。

“渐变条颜色1”属性设置每个菜单左边渐变条最左边的颜色,并且顶级菜单当被选中后该颜色为顶级菜单渐变颜色最顶边的颜色。

“渐变条颜色2”属性设置每个菜单左边渐变条最右边的颜色,并且顶级菜单当被选中后该颜色为顶级菜单渐变颜色最底边的颜色。

“菜单条点燃颜色1”属性为顶级菜单被点燃时的渐变颜色最顶边的颜色。

“菜单条点燃颜色2”属性为顶级菜单被点燃时的渐变颜色最底边的颜色。


“点燃区边框颜色”属性设置当菜单项目被点燃时边框的颜色。

“分割条颜色”属性设置分割条的颜色。

3. “图片组”属性

该属性存放一个供菜单显示图标时使用的图片组文件。

4. “图片索引”属性

图片索引属性设置每个菜单项左侧显示的图片索引,图片的索引值为该图片在图片组中的位置,从0开始,0代表图片组中的第一个图片。点击该属性上的“”按钮,会弹出图片索引管理对话框,如图5-68所示。

该对话框中列出了窗口中的所有菜单项,可以选择指定的菜单项设置图片索引。在该属性上点击鼠标右键,会弹出右键菜单,菜单中有“删除内容”和“写到文件”选项。选择“删除内容”则会将当前的设置全部删除;选择“写到文件”选项会将当前的图片索引设置保存到一个文件中,在代码中可以使用“读入文件()”命令将保存后文件读入然后直接赋值给“图片索引”属性,这样可以直接使用保存的设置,例如:

超级菜单 1. 图片索引 = 读入文件 (“C:\索引保存文件”)

【注意】保存图片索引文件时可以不设置扩展名。使用该方法可以将不同的设置方案保存到不同的文件中,程序中就可以随意在多个设置方案中转换了。

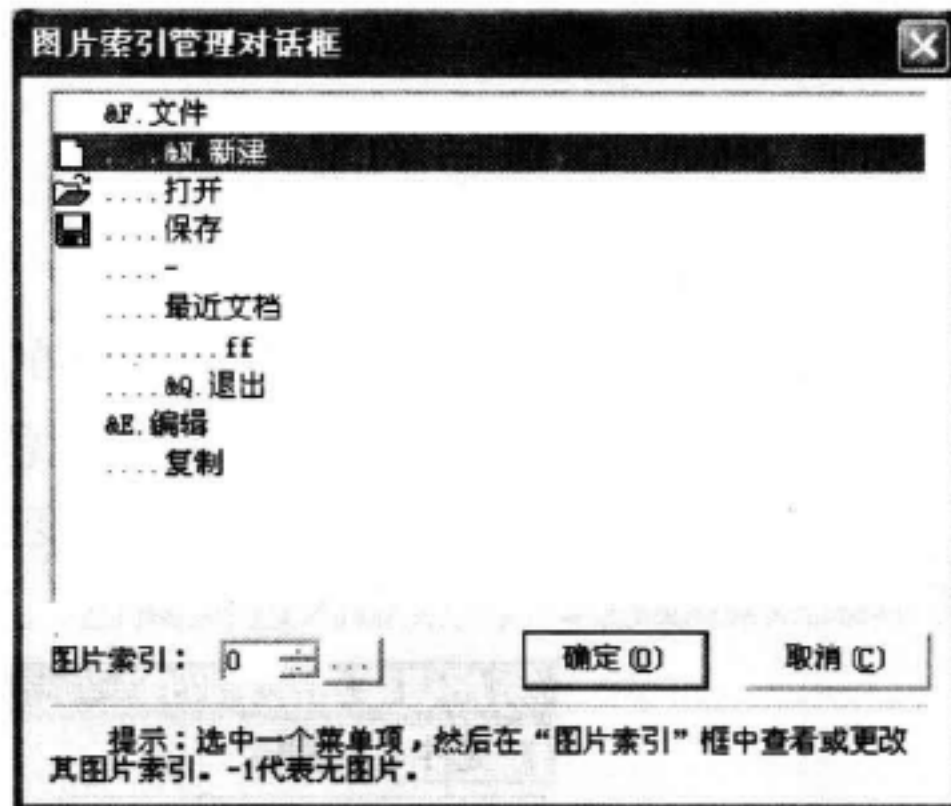


图 5-68 图片索引管理对话框

5.3.2 超级菜单的方法

“置图片()”方法

该方法用来设置某个菜单项的标志图片,即在程序运行前没有在“图片索引”属性中设置,在程序后便可以使用“置图片()”方法设置某菜单项左边的小图标。

该方法有2个参数,第1个参数填写要设置的菜单项,第2个参数填写要设置图片索引。

例如将“新建”菜单的图片索引设置为2:

超级菜单 1. 置图片 (新建, 2)

5.3.3 超级菜单的事件

1. “被点燃”事件

某菜单项被点燃时触发该事件,即当鼠标停留在某菜单上时触发该事件,该事件有1个参数,为被点燃的菜单项。

有了“被点燃”事件,设置菜单的动态提示文本就很方便了,具体参见例程5-31,例程中“被点燃”事件代码如图5-69所示。

程序中判断被点燃的菜单后,改变状态条中的文本,可以用来提示被选择菜单项的作用,效果如图5-70所示。

2. “被关闭”事件

当所有弹出的菜单被关闭(从屏幕上消失)时触发该事件。

例如,当所有菜单被关闭则将状态条文本设置为“就绪”,代码如下:

状态条 1. 置文本(0,“就绪”)

子程序名	返回值类型	公开	备注		
超级菜单1_被点燃					
参数名	类型	参考	可空	数组	备注
被点燃的菜单项	菜单				

判断 (被点燃的菜单项 = 新建)

状态条1.置文本 (0, "新建一个文件")

判断 (被点燃的菜单项 = 打开)

状态条1.置文本 (0, "打开一个已存在的文件")

判断 (被点燃的菜单项 = 保存)

状态条1.置文本 (0, "保存当前正在编辑的文件")

判断 (被点燃的菜单项 = 最近文档)

状态条1.置文本 (0, "打开最近编辑过的文档")

判断 (被点燃的菜单项 = 退出)

状态条1.置文本 (0, "退出本程序")

图 5 - 69 “被点燃”事件运行代码

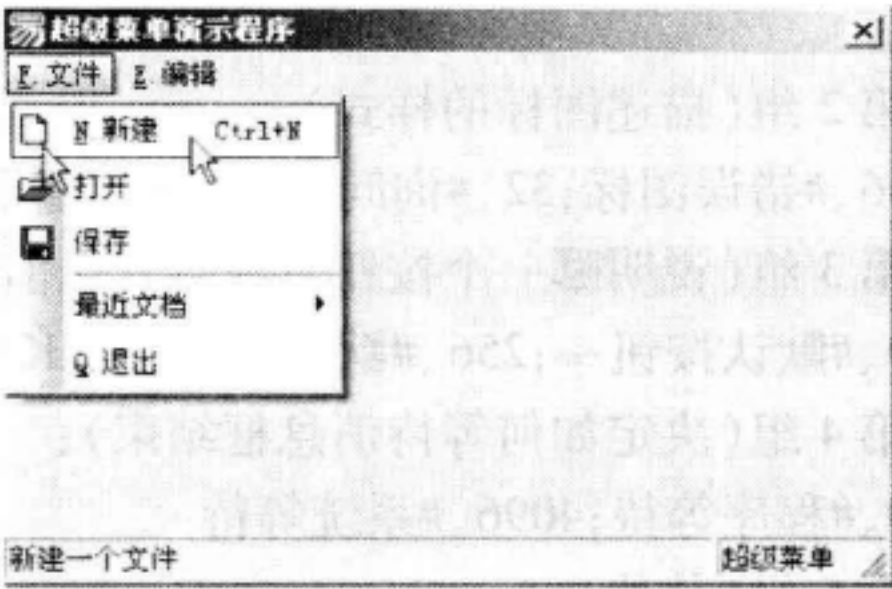


图 5 - 70 状态条显示菜单提示文本

5.4 信息框组件

5.4.1 信息框概述

信息框是常用到的提示信息命令。在此,将它的参数意义说明如下。一个简单的提示信息框的程序代码如下,运行后效果如图 5 - 71 所示。

信息框(“数据库还未能打开”,0,“请注意!”)

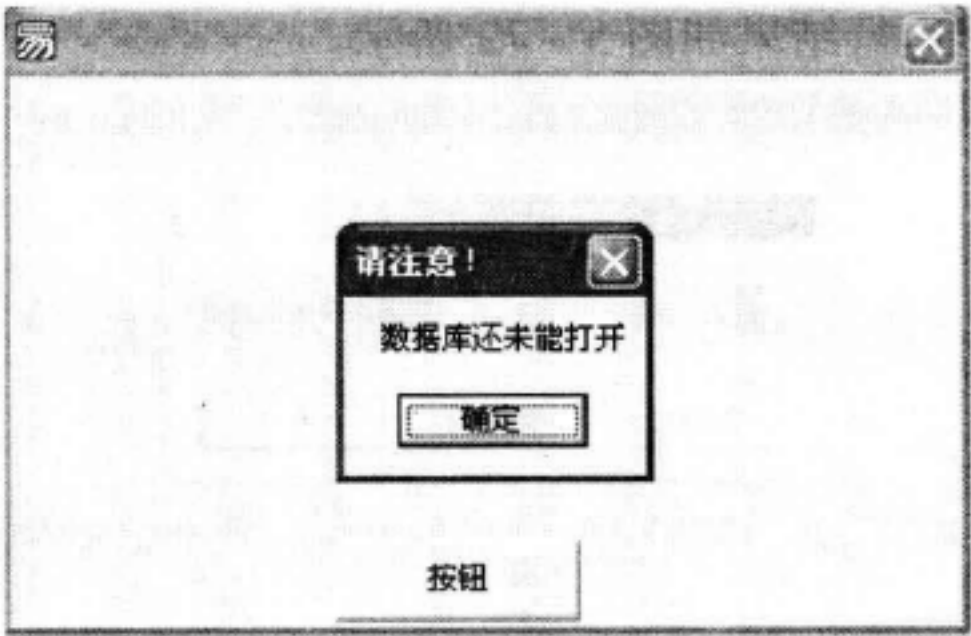


图 5 - 71 简单提示信息框运行结果

信息框一共有 3 个参数,第 1 个参数是在信息框中显示的内容。相当于窗口组件中放置了一个标签组件,并且显示标签组件的标题。

第 2 个参数是数值型,表示显示的按钮。相当于窗口内放置了一按钮组件的标题属性。

第 3 个参数是信息框窗口提示。如果省略,默认为文本“信息:”。相当于窗口属性中的标题。

具体参见例程 5 - 32。

5.4.2 信息框提示按钮形态

第 2 个参数非常重要。将光标定位在代码行上按 F1 键,“提示”子夹中会显示如下的信息:

参数 <2> 的名称为“按钮”,类型为“整数型(int)”,初始值为“0”。参数值由以下几组常量值组成,在将这些常量值相加以生成参数值时,每组值只能取用一个数字(第 5 组除外):

第1组(描述对话框中显示按钮的类型与数目):

0、#确认钮;1、#确认取消钮;2、#放弃重试忽略钮;3、#取消是否钮;4、#是否钮;5、#重试取消钮

第2组(描述图标样式):

16、#错误图标;32、#询问图标;48、#警告图标;64、#信息图标

第3组(说明哪一个按钮是缺省默认值):

0、#默认按钮一;256、#默认按钮二;512、#默认按钮三;768、#默认按钮四

第4组(决定如何等待消息框结束):

0、#程序等待;4096、#系统等待

第5组(其他):

65536、#位于前台;524288、#文本右对齐

由以上信息可以看到,信息框按钮的基本形态有6种,就是第1组所说明的。信息框中还有4种不同的图标,就是第2组所说明的。第3组是说明焦点会在哪一个按钮上。第4组是决定如何等待消息框结束的方式,有程序等待方式与系统等待方式两种。第5组是位于前台与文本右对齐的功能值。

以上从第1组到第4组,对于想要的可以挑选其中的一种相加得到。例如:想要“放弃重试忽略钮”,显示“询问图标”,默认按钮为第3个,程序等待,文本右对齐。这样,就可以将上述值相加“ $2 + 32 + 512 + 0 + 524288$ ”即填充到第2个参数中。

信息框(“数据库还未能打开”, $2 + 32 + 512 + 0 + 524288$,”请注意!”)

具体参见例程5-33。运行结果如图5-72所示。

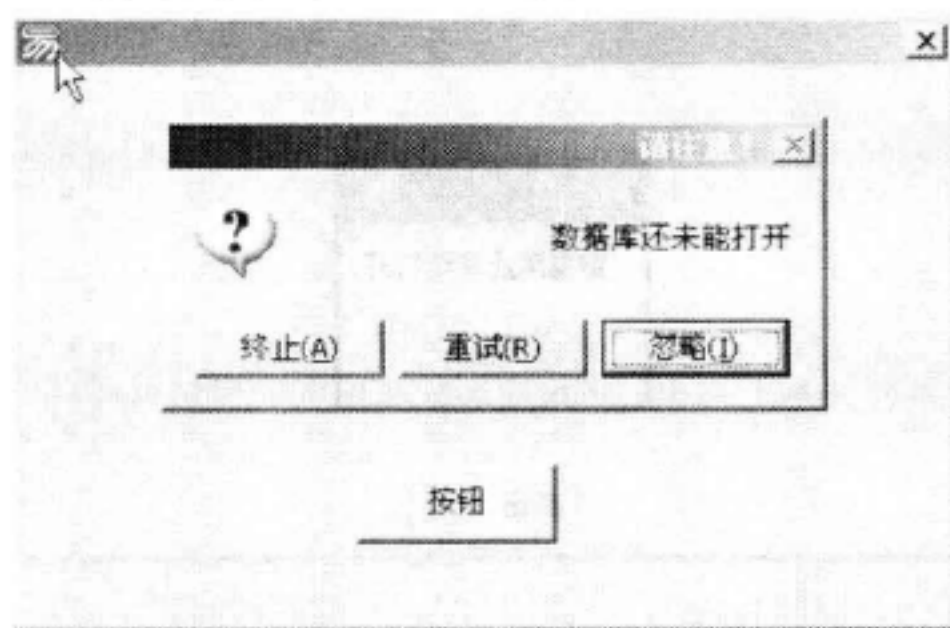


图5-72 信息框例程运行结果

使用一个循环来测试以上的用途,代码如下:

计次循环首(70, 1)

 信息框(“数据库还未能打开”, 1, “请注意!” + 到文本(I))

计次循环尾()

具体参见例程5-34。

当然,如果记不住,或使程序更加直观,可以直接将常量直接引用到程序代码中,如:

信息框(“字段名称不能为空”, #错误图标, “错误”)

或:

信息框(“字段名称不能为空”, #确认取消钮 + #错误图标 + #默认按钮二, “错误”)

5.4.3 信息框的返回值

信息框的返回值说明如下:在对话框中显示信息,等待用户单击按钮,并返回一个整数告诉用户单击哪一个按钮。该整数为以下常量值之一:0. #确认钮;1. #取消钮;2. #放弃钮;3. #重试钮;4. #忽略钮;5. #是钮;6. #否钮。如果对话框有“取消”按钮,则按下 Esc 键与单击“取消”按钮的效果相同。

将上述例程 1 改一下,加一个标签组件,命令行代码为:

标签 1. 标题 = 到文本 (信息框 (“数据库还未能打开”, 3, “请注意!”))

试运行,当单击某个按钮后,标签就接受到了按钮的信息,从而返回相应的数值,而程序接收到相应的数值,就可以按不同的结果进行判断操作。

具体参见例程 5-35,例程运行结果如图 5-73 所示。

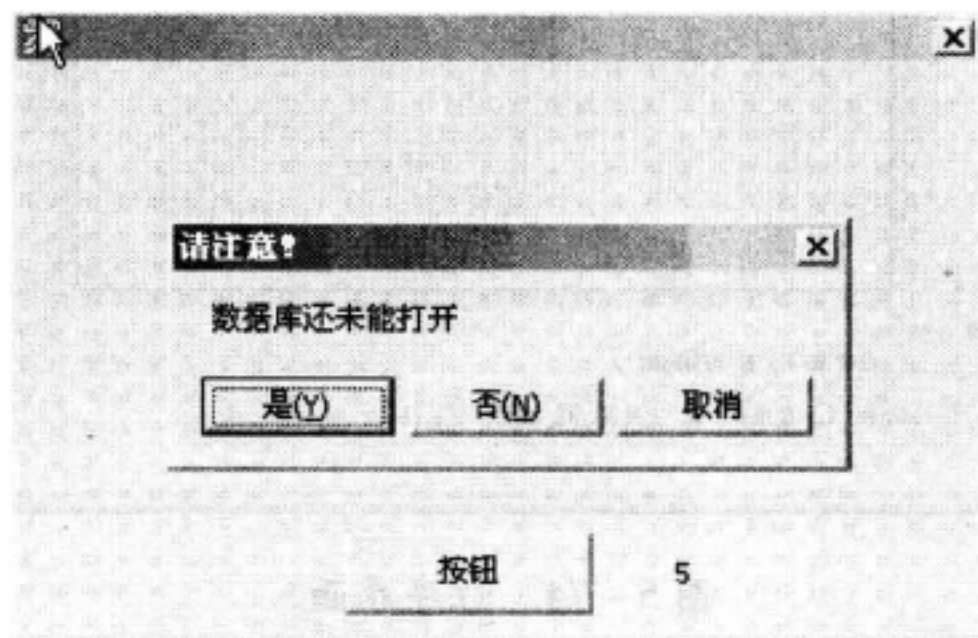


图 5-73 信息框例程运行结果

实际上,可以使用一个变量返回信息框的值。然后与常量显示比较,代码可以这样写:

子程序:_按钮 1_被单击

局部变量:临时变量 数据类型:整数型

临时变量 = 信息框 (“请确认是否继续运行!”, 3, “请注意!”)

判断 (临时变量 = #是钮)

 标签 1. 标题 = “按下是钮.”

判断 (临时变量 = #否钮)

 标签 1. 标题 = “按下否钮.”

判断 (临时变量 = #取消钮)

 标签 1. 标题 = “按下取消钮.”

默认

判断结束

试运行该程序,只要在信息框弹出时,单击某一个按钮,就会返回一个按钮的值。

具体参见例程 5-36。

5.5 输入框组件

如果只输入一个简单的数据,就不需要太多的窗口设计,使用输入框命令行组件就可以完

成。输入框相当于创建一个动态窗口,窗口的标题和提示标签可以修改,并可以返回一个值,还可以将初始信息写在类似于一个编辑框中。上述的内容只需一行命令就能完成。

5.5.1 输入框操作

输入框组件可以接收用户输入的数据,并且保存在某一个变量中供程序使用。新建一个新的易程序,在启动窗口中放置两个组件,一个标签组件和一个编辑框组件,如图 5-74 所示。

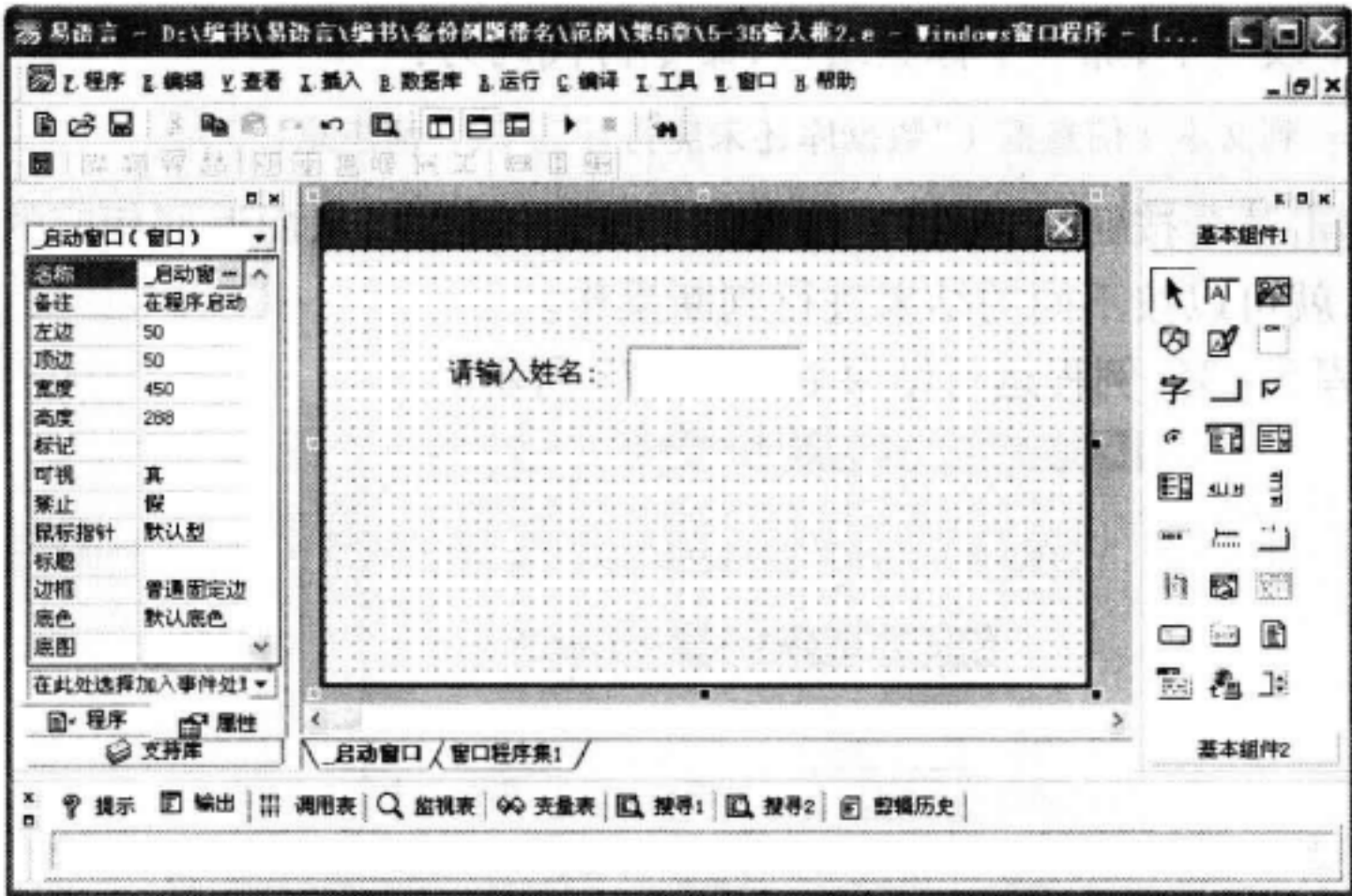


图 5-74 程序界面

选中标签组件,将标题改为:“请输入姓名:”,使用属性面板中的事件下拉菜单,选中“鼠标左键被按下”,生成“_标签 1_鼠标左键被按下”事件子程序。

输入框(“请输入姓名”,“基本信息”, , 存姓名, 1)

编辑框 1. 内容 = 存姓名

其中,如果出现“存姓名”不存在的情况时,可以新增为程序集变量,并且设置为文本型,如图 5-75 和图 5-76 所示。

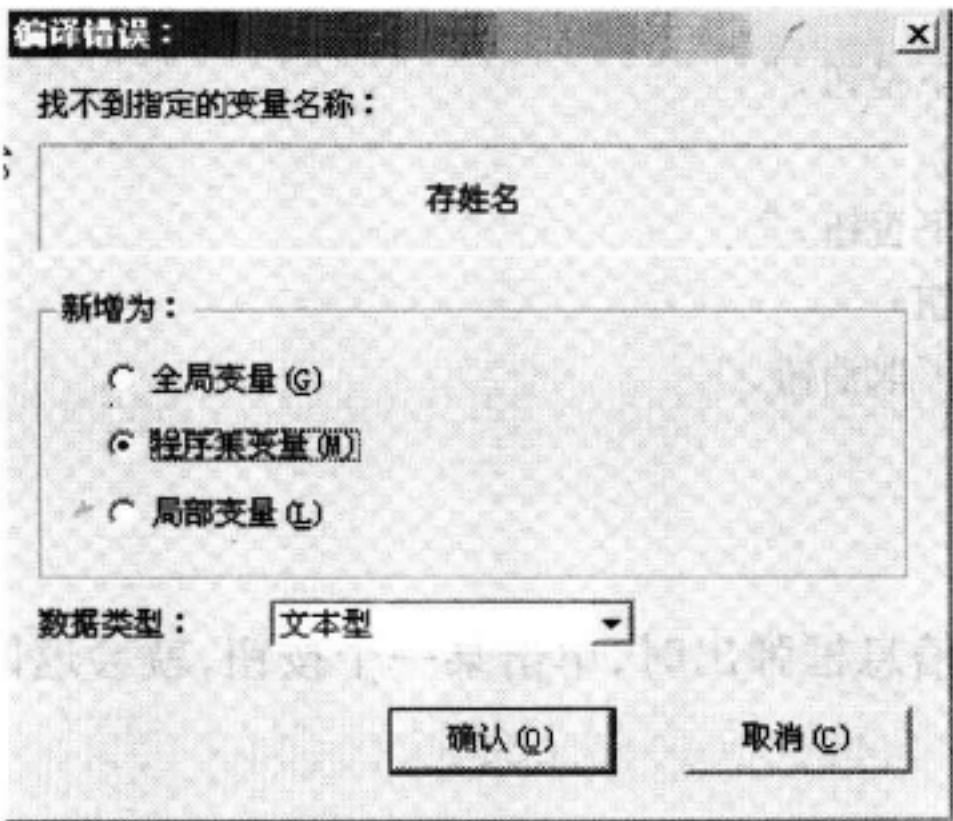


图 5-75 将“存姓名”新增为程序集变量

单击标签就会弹出输入框组件的界面,输入的数据还会显示在编辑框中,如图 5-77 所示。

小香雪

竹裏尋幽徑

梅間卜野居

功德山區聯

天池

淥水入澄照

青山猶古姿

倚虹園匾聯

致佳樓

含多行,可在各行之间用回车符(即“字符(13)”)、换行符(即“字符(10)”)或回车换行符的组合(即:“字符(13)+字符(10)”)来分隔。如果提示信息太长或行数过多,超过部分将不会被显示出来。

参数<2>的名称为“窗口标题”,类型为“文本型(text)”,可以被省略。参数值指定显示在对话框标题栏中的文本。如果省略,默认为文本“请输入:”。

参数<3>的名称为“初始文本”,类型为“文本型(text)”,可以被省略。参数值指定初始设置到对话框输入文本框中的内容。

参数<4>的名称为“存放输入内容的变量”,类型为“通用型(all)”,提供参数数据时只能提供变量。参数值所指定的变量可以为数值或文本型,用于以不同的数据类型取回输入内容。

参数<5>的名称为“输入方式”,类型为“整数型(int)”,可以被省略。参数值可以为以下常量值:1、#输入文本;2、#输入整数;3、#输入小数;4、#输入密码。如果省略本参数,默认为“#输入文本”。

5.5.3 输入框输入编程

若将上述程序改为密码方式,要将编辑框的“输入方式”属性改为“密码输入”,如图5-78所示。

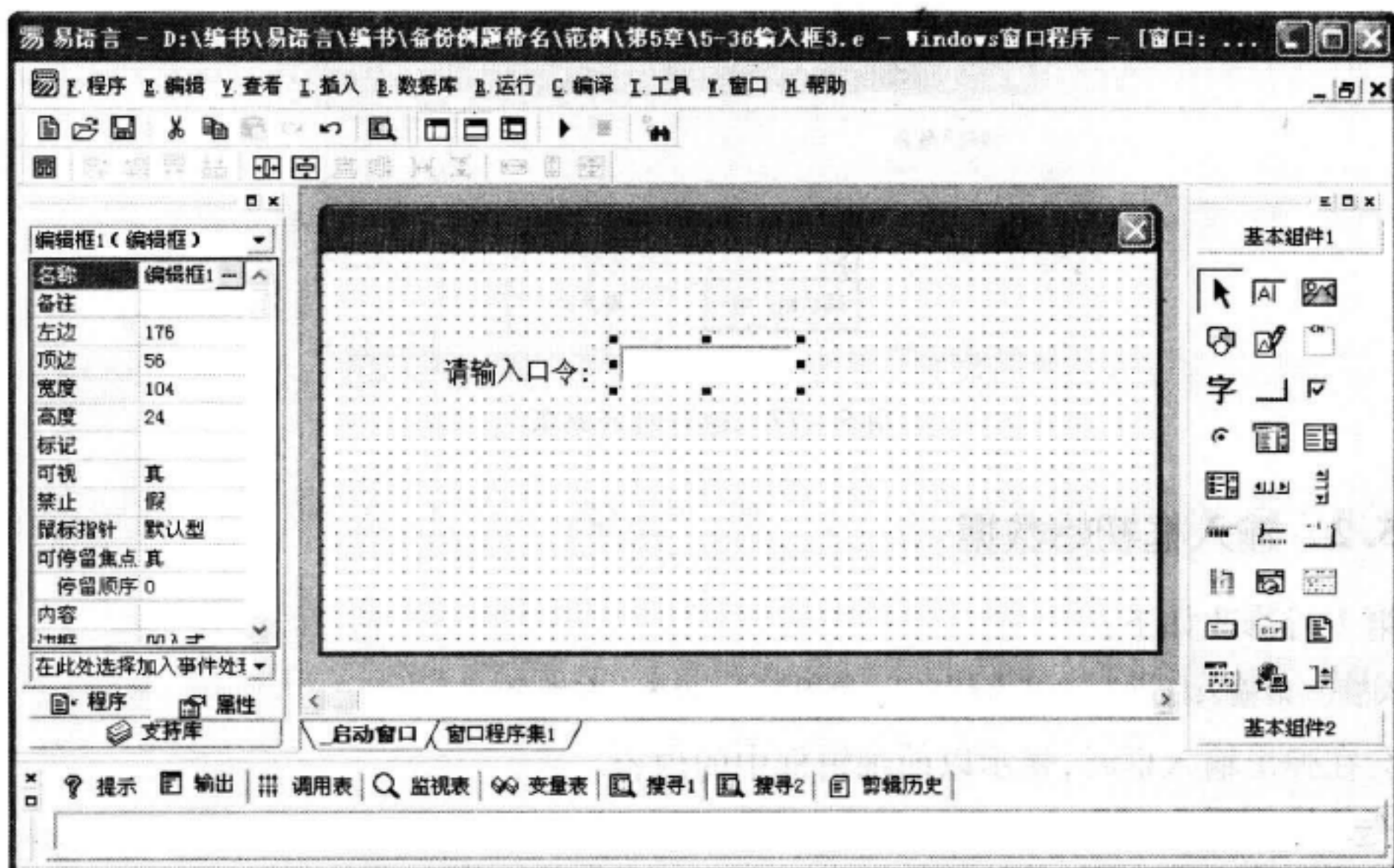


图 5-78 编辑框的“输入方式”属性改为“密码输入”

将命令行改为:

输入框(“请输入姓名”,“基本信息”,编辑框 1. 内容,存姓名,4)

编辑框 1. 内容 = 存姓名

输入框的最后一个参数为 4。输入密码时,只会显示星号,而不会显示原文本,如图 5-79 所示。

具体参见例程 5-38。

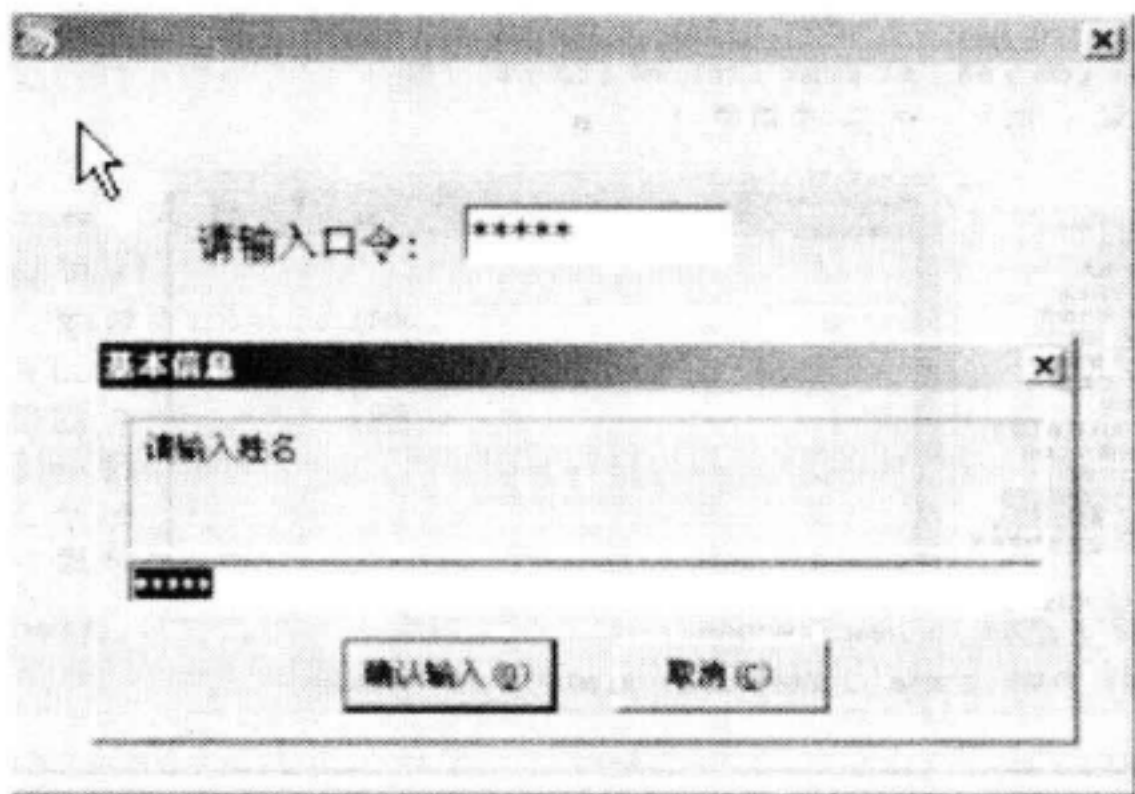


图 5-79 输入文本显示为星号

5.6 数据库维护

当打开一个易语言的数据库时,如果不想专门为这个数据库编写基本的操作方法,如“到上一记录”,“到下一记录”,“增加记录”等操作。可以直接使用一个命令,实现基本的维护,如图 5-80 所示。

编辑(,,)

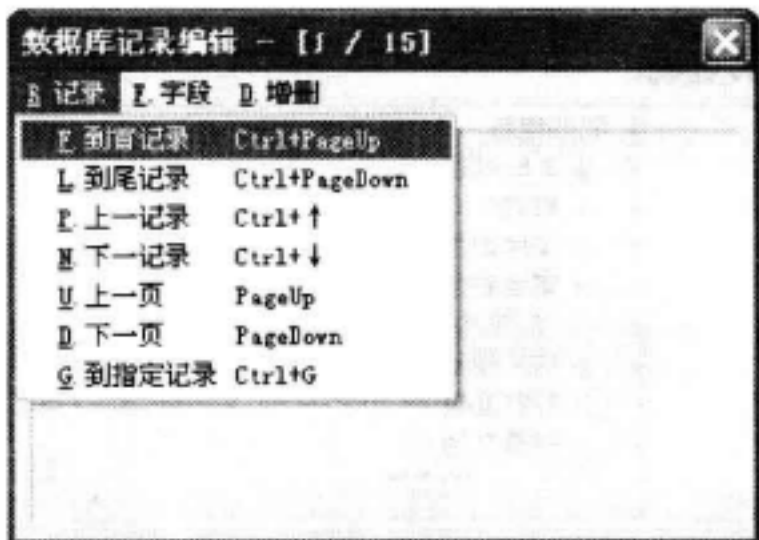


图 5-80 使用“编辑”命令后的样子

在已打开数据库的情况下,就可对当前数据库进行维护。如果当前没有数据库打开,可以使用相关的程序打开。

如果在当前数据库打开的情况下,想查看另一个数据库,可以保存当前数据库的名称及指针,当操作另一个数据库结束时重新打开以前的数据库名称及指针。

具体参见例程 5-39。

5.7 浏览文件夹

此外,在高版本中还支持“浏览文件夹”命令行组件,具体参见例程 5-40。

可以看一下支持库面板的“操作系统界面功能支持库”中有无这个命令:“浏览文件夹”。自己编写一段程序试验一下。新建易程序后,放两个编辑框及两个按钮,如图 5-81 所示。

分别双击这两个按钮,输入以下代码:

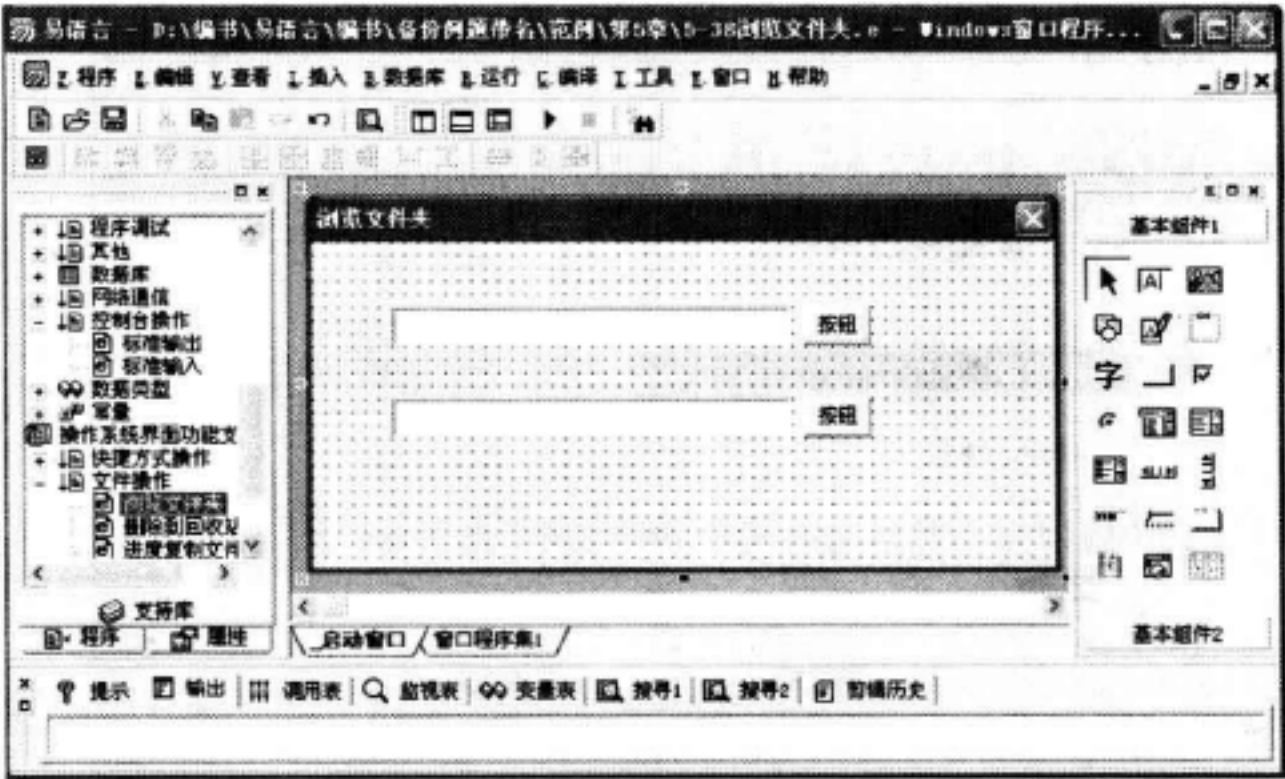


图 5-81 设计程序界面

子程序:_按钮 1_被单击
编辑框 1. 内容 = 浏览文件夹(“请确定文件位置”,)

子程序:_按钮 2_被单击
编辑框 2. 内容 = 浏览文件夹(“请选择一个文件”, 真)

运行后,就可以看到效果了,如图 5-82 所示。

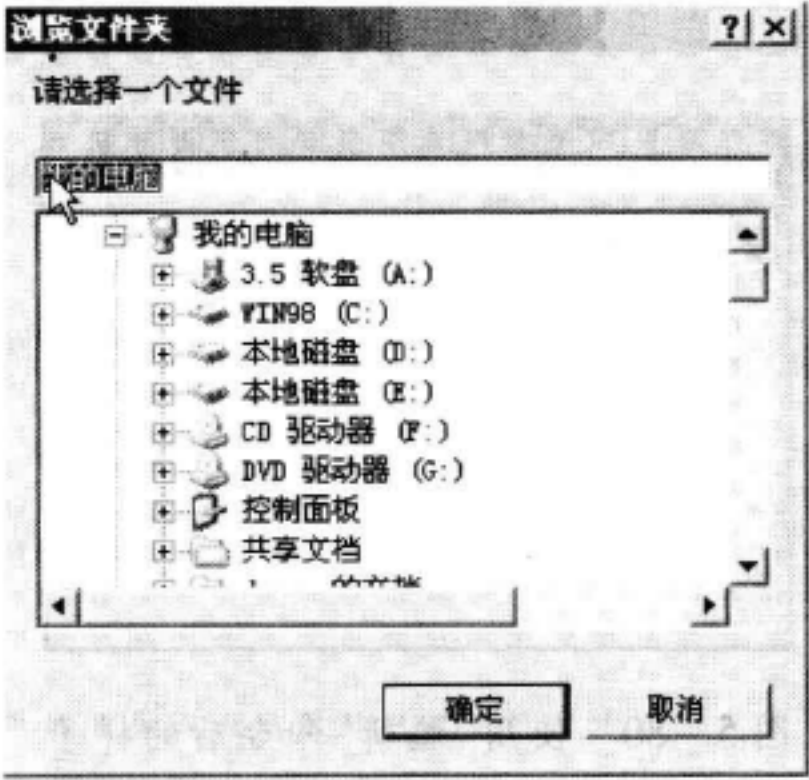



图 5-82 运行结果

第 6 章 鼠标触发组件

6.1 按钮组件

6.1.1 按钮组件概述

按钮  通常用于向计算机发送指令(事件驱动)、执行命令的接口。比如在程序运行时,用户单击按钮便可以触发事件并执行预定的程序功能。按钮是最常用的组件,同时也是最典型的组件,学习其他组件之前应先学好按钮组件的用法。

6.1.2 按钮的重要属性

按钮的重要属性有“名称、标题、禁止、可视、类型、图片、字体”等,分别介绍如下。

(1) “名称”属性是按钮的唯一标识。对任何组件的操作都要指定操作对象的名称。名称是在程序内部引用的代号,因此,一定不能重复,并且有可识别的取名方法。名称在程序中是无法改变的,因此,在程序设计时要规定好名称。一个窗口中有 4 个按钮,它们的标题可以一样,但名称不能一样,如图 6-1 所示窗口的 4 个按钮组件的标题一样,都叫按钮,在编程状态下,试着用鼠标激活每一个按钮,再查看属性面板中的“名称”属性,可以看到它们的名称都不一样。

(2) “标题”属性用于显示按钮上的文字。多个按钮的标题可以重复,标题与名称的不同在于标题只是一种说明文字,而并非程序所引用的关键字。因此,如果出现多个标题相同的按钮就不足为奇了,而名称却不可以重复。可以通过下面这个命令行在程序中将“标题”属性改变:

按钮 1. 标题 = “我是新的标题”

运行后结果如图 6-2 所示。

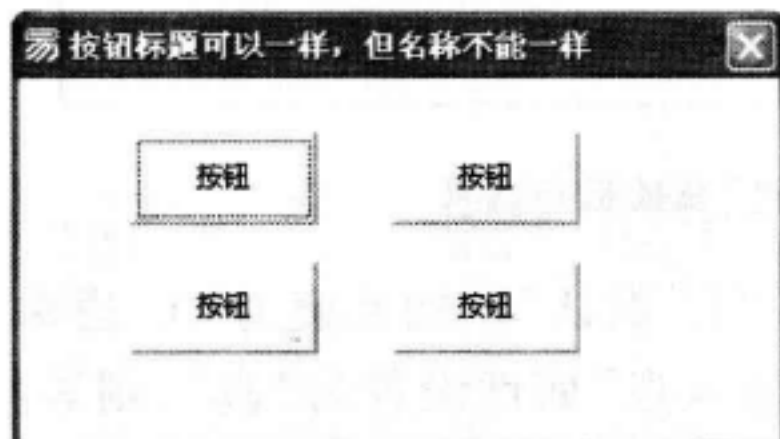


图 6-1 按钮的名称属性



图 6-2 按钮的标题属性

(3) “禁止”属性可控制按钮是否能操作。其值是逻辑值,只能为“真”或“假”。默认情况下为“假”,即可被看到。如果为“真”时,按钮上的文字也变为灰色,并且不可操作。也可以通过以下命令在程序中将这个属性改变:

按钮. 禁止 = 假

如图 6-3 所示,按钮 1 是被禁止了的按钮,其上相应的文字也变为灰色,按钮 2 则是没有被禁止的按钮。

(4) “可视”属性可控制按钮是否能被看到。其值是逻辑值,只能是“真”或“假”。默认情况下为“真”,即可被看到。可以通过以下命令在程序中将这个属性改变:

```
按钮.可视 = 假
```

【范例 6-1】按钮的可视属性演示,设置按钮 1 的可视属性是“真”,按钮 2 的可视属性为“假”,点击按钮 3 实现按钮 1 和按钮 2 全部可视,点击按钮 4 实现按钮 1 和按钮 2 全部隐藏。具体参考例程 6-1。

具体的程序代码如图 6-4 所示。

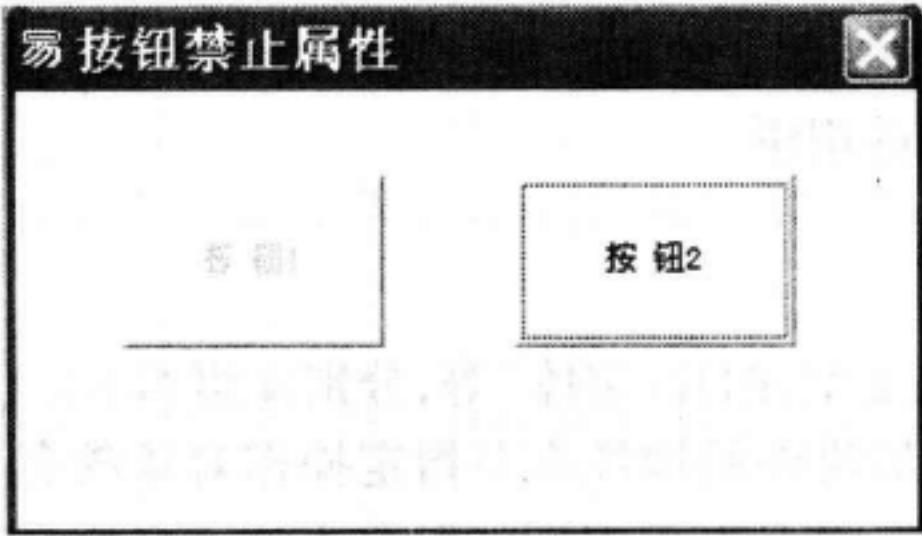


图 6-3 按钮的“禁止”属性

窗口程序集名	保留	保留	备注
窗口程序集1			

子程序名	返回值类型	公开	备注
_按钮3_被单击			

按钮1.可视 = 真
按钮2.可视 = 真

子程序名	返回值类型	公开	备注
_按钮4_被单击			

按钮1.可视 = 假
按钮2.可视 = 假

图 6-4 按钮可视属性运行代码

【运行结果】运行例程 6-1,观测分别单击按钮 3、按钮 4 后按钮 1 和按钮 2 的变化,如图 6-5 所示。

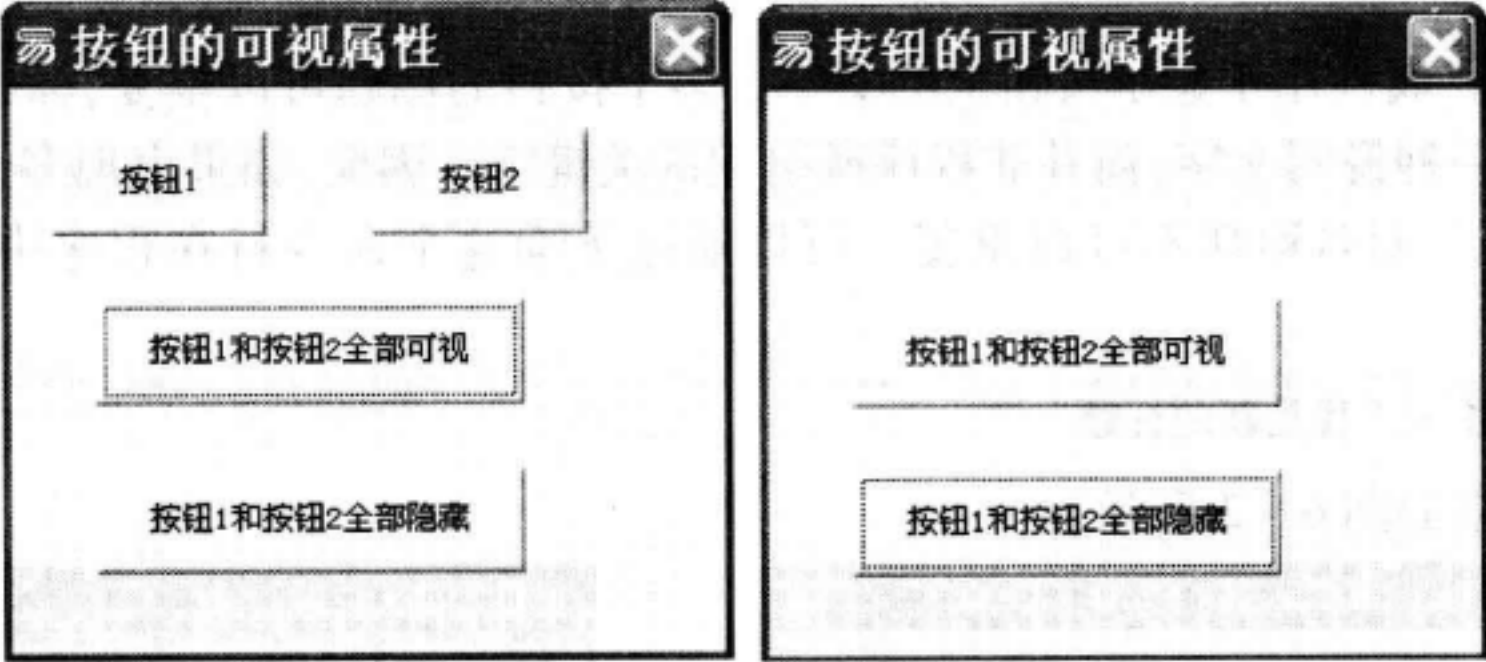


图 6-5 按钮“可视”属性运行结果

(5) “类型”属性有两个可选值:“0. 通常”、“1. 默认”。初始值为“0. 通常”。当在窗口中按下回车键时,如果没有将窗口的“回车下移输入点”属性设置为“真”,则等同于按下了具有“默认”类型的按钮。因此,具有“默认”类型的按钮在同一窗口中应当只有一个。“通常”类型的按钮无对应的操作默认键。“类型 = 通常”与“类型 = 默认”的按钮在外观上有一些不同。可以通过命令在程序中将这个属性改变。例如:

```
按钮.类型 = 0
```

(6) “图片”属性可指定显示于按钮上的图片,支持 BMP、JPEG、GIF、ICO、CUR 等格式。如果设置了本属性,则按钮的“标题”属性无效,即图片将覆盖按钮上的文字。如果既想要图

片,又想要文字呢? 可以做一个带文字的图片覆盖在按钮上。可以通过以下命令在程序中将这个属性改变:

```
按钮. 图片 = 读入文件("d:\风景.jpg")
```

【范例 6-2】用三种方式实现按钮“图片”属性的设置,一是在按钮的属性面板中直接加入图片,二是为按钮加入资源中的图片,三是为按钮加入任意的图片。具体参考例程 6-2。例程代码如图 6-6 所示。

【运行结果】运行例程 6-2,观测单击右侧两个按钮后左侧 3 个按钮的变化,如图 6-7 所示。

窗口程序集名	保留	保留	备注
窗口程序集1			

子程序名	返回值类型	公开	备注
_按钮4_被单击			

按钮2. 图片 = #按钮图片

子程序名	返回值类型	公开	备注
_按钮5_被单击			

如果真 (通用对话框1. 打开 0 = 真)
按钮3. 图片 = 读入文件 (通用对话框1. 文件名)

图 6-6 按钮图片属性源代码



图 6-7 按钮图片属性运行结果

(7) “字体”属性可指定按钮标题文字的字体。注意文字颜色始终是黑色的,无法改变。在程序中可使用下述代码来控制“字体”属性。

例如,设计按钮标题文字是宋体、加粗、倾斜程序代码为:

```
按钮. 字体. 字体名称 = “宋体”
```

```
按钮. 字体. 加粗 = 真
```

```
按钮. 字体. 倾斜 = 真
```

(8) “标记”属性可记录组件的附加文本信息。类型是文本型,可以用来保存一个变量信息,节约变量的定义与使用。

如图 6-8 所示,在最下排的“点击后,改变按钮 1 的标题属性”按钮中的代码如下:

```
按钮 1. 标记 = “已修改过了!”
```

```
标签 1. 标题 = 按钮 1. 标记
```

程序运行后,就可以改变按钮的“标记”属性了。

【注意】这个“标记”属性基本上等同于一个全局的变量,所以使用起来十分方便。如果在其他窗口引用,可以使用“窗口. 按钮. 标记”进行引用。

(9) “可停留焦点”属性

在使用 Tab 键或光标键在各组件之间移动焦点时,“可停留焦点”属性可控制是否允许在本组件上停留。它的值只能为“真”或“假”,默认为“真”。如果组件的“可停留焦点”属性为“真”,则组件还有另外一个属性“停留顺序”控制焦点停留的顺序,即窗口上各组件获得输入焦点的顺序。

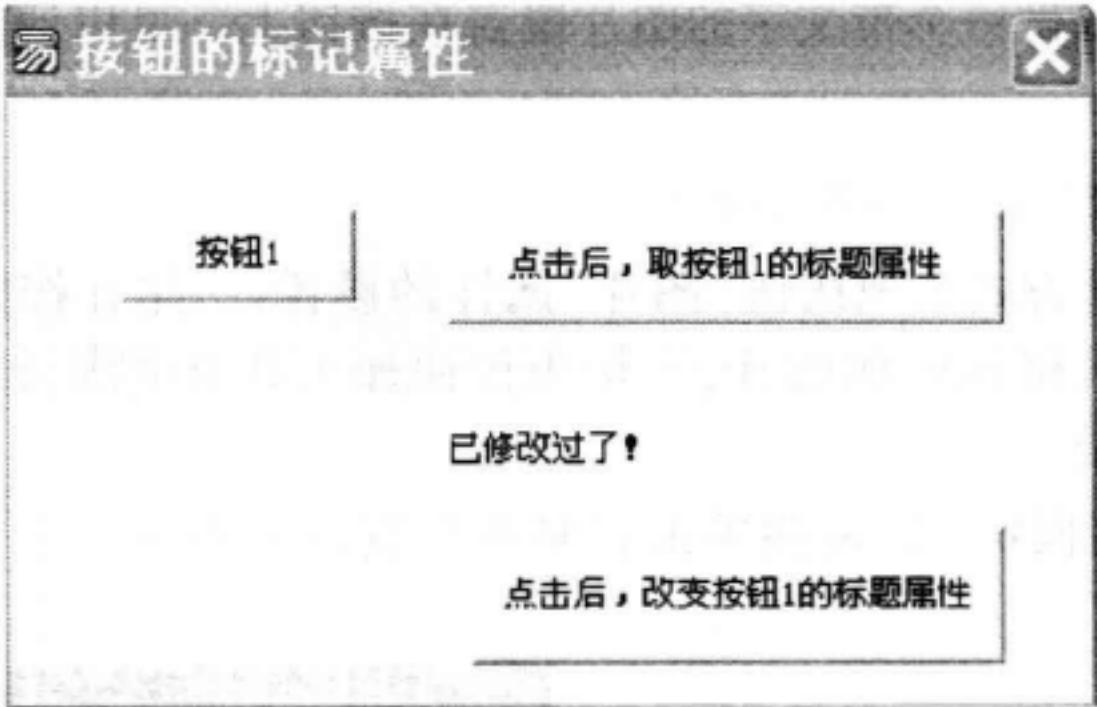


图 6-8 按钮的标题属性

6.1.3 按钮的专有方法

按钮没有专有方法。展开支持库面板，单击“系统核心支持库”中的“数据类型”→“按钮”，在组件前有“+”号的表示有下级分支，可以继续展开。如图 6-9 所示，可以看出按钮组件无专有方法。

6.1.4 按钮的重要事件

激活一个按钮后，就可以在属性面板的最下一排中找到一些事件。按钮的事件有 12 种，如图 6-10 所示。单击列表中的某一事件就会产生相应的事件子程序，并且可以进入事件子程序的设计界面。

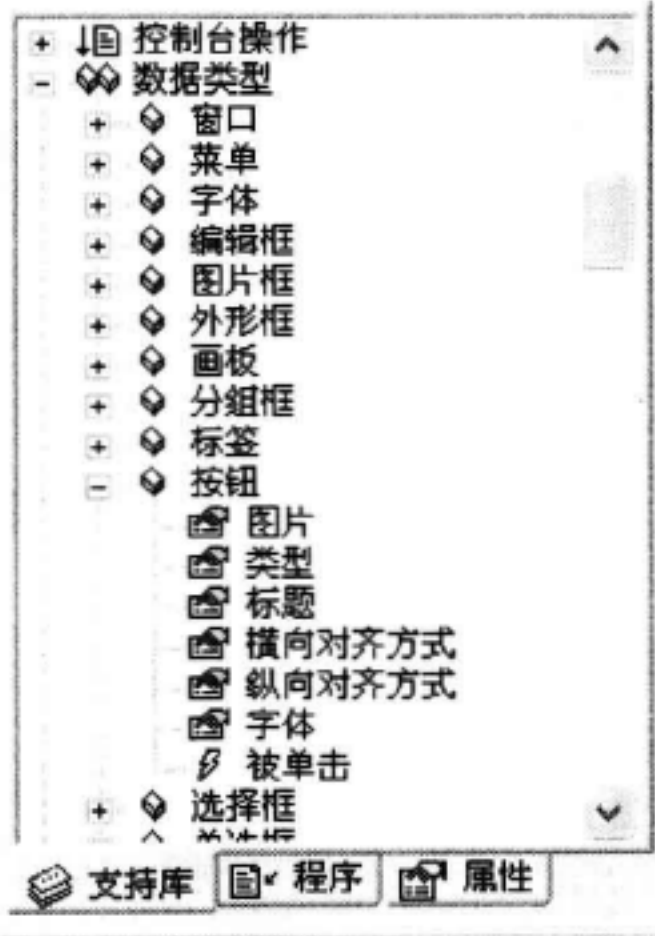


图 6-9 按钮组件无专有方法



图 6-10 按钮的重要事件

下面分别介绍按钮组件的以下事件：

- (1) “被单击”事件的产生时机：当按钮被鼠标左键单击时产生本事件。按钮“被单击”事件十分常用，前面章节也已多次使用了该事件。当按钮组件排列在窗口中时，可以双击按钮进入“被单击”事件，或者激活按钮后，单击属性面板下方的事件下拉菜单，选中“被单击”即可生成某按钮的“被单击”事件。
- (2) “鼠标左键被按下事件”的产生时机：当鼠标左键被按下时触发该事件。

- (3) “鼠标右键被按下事件”的产生时机:当鼠标右键被按下时触发该事件。
- (4) “被双击事件”的产生时机:当双击鼠标按键时,左键或右键都可以产生该事件。
- (5) “按下某键”事件可以响应键盘操作,“放开某键”事件时机比“按下某键”事件迟一些。

【范例6-3】设置4个按钮的对应事件分别为“被单击”、“鼠标左键被按下”、“鼠标右键被按下”、“被双击”,当触发该事件,对应按钮的标题分别为:“被单击”、“本按钮鼠标左键被按下”、“鼠标右键被按下”、“本按钮已被双击”。具体参考例程6-3。

例程代码如图6-11所示。

窗口程序集名	保留	保留	备注
窗口程序集1			

子程序名	返回值类型	公开	备注		
_按钮_鼠标左键被按下	逻辑型				
参数名	类型	参考	可空	数组	备注
横向位置	整型				
纵向位置	整型				
功能键状态	整型				

按钮.标题 = “本按钮鼠标左键被按下!”

子程序名	返回值类型	公开	备注		
_按钮4_被双击	逻辑型				
参数名	类型	参考	可空	数组	备注
横向位置	整型				
纵向位置	整型				
功能键状态	整型				

按钮4.标题 = “本按钮已被双击!”

子程序名	返回值类型	公开	备注		
_按钮6_鼠标右键被按下	逻辑型				
参数名	类型	参考	可空	数组	备注
横向位置	整型				
纵向位置	整型				
功能键状态	整型				

按钮6.标题 = “鼠标右键被按下!”

子程序名	返回值类型	公开	备注
_单击按钮_被单击			

单击按钮.标题 = “被单击!”

图6-11 按钮事件源代码

【运行结果】运行例程6-3,观测分别单击4个按钮后各自的变化,如图6-12所示。

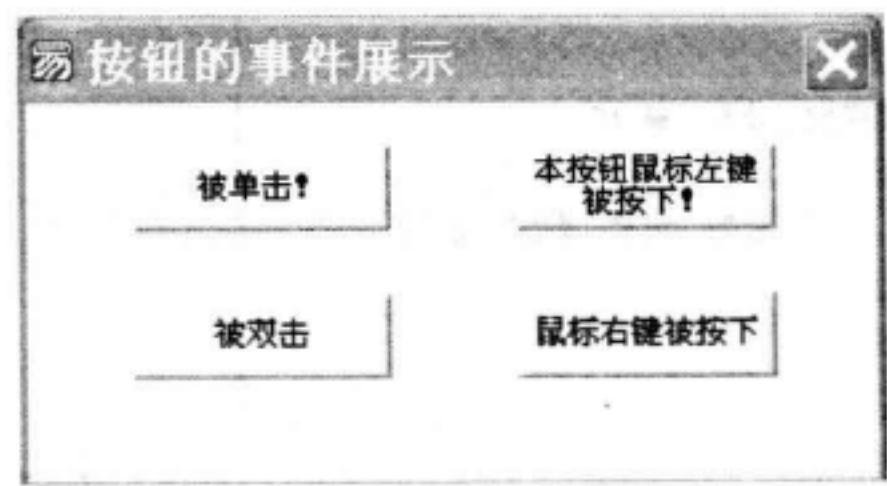


图6-12 按钮的事件演示

【注意】单击左排第 1 个按钮,按钮的标题被改变了。单击右排第 1 个按钮后,按钮的标题也被改变了。虽然都是单击操作,但事件触发时机是不同的,“鼠标左键被按下”事件比“被单击”事件的触发实际早。为了测试,可以单击右排的第 1 个按钮,体会当按钮被按下时即产生“鼠标左键被按下”事件,而当单击完成后才会产生“被单击”事件。

6.1.5 按钮的提示信息

鼠标停留在按钮上时,就会提示相应的信息。下面通过一个例子来看按钮的提示信息如何使用。

【范例 6-4】新建一个易程序,加入一个按钮组件和一个标签组件。鼠标停留在按钮上时,就提示相应的信息:“这是一个按钮”。具体参考例程 6-4。

标签组件的属性设置如图 6-13 所示。

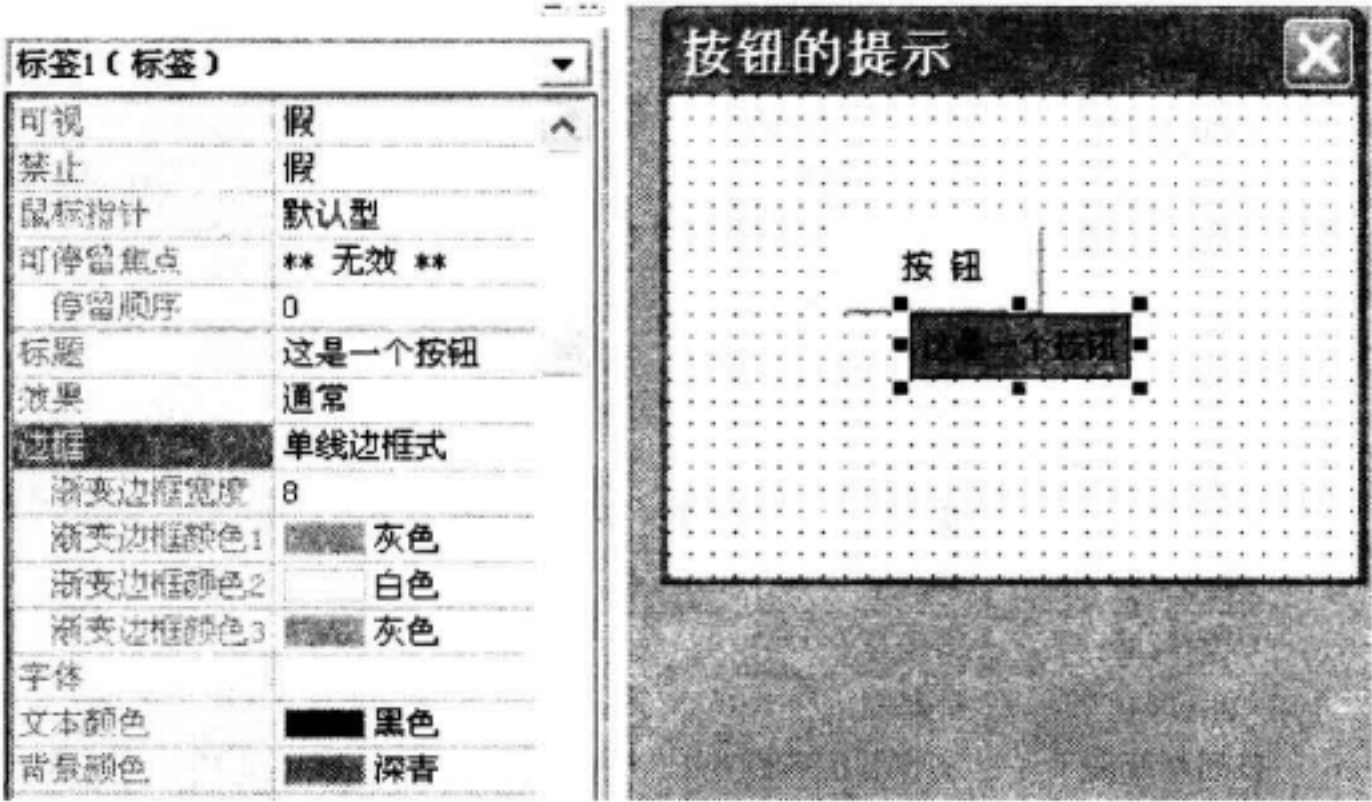


图 6-13 标签的属性设置

选中按钮组件后,在属性面板下方的事件下拉菜单中选“鼠标位置被移动”后进入程序设计界面,具体的程序代码如图 6-14 所示。

【运行结果】运行例程 6-4,观测鼠标在窗体中移动时的变化,如图 6-15 所示。

窗口程序集名	保留	保留	备注
窗口程序集1			

子程序名	返回值类型	公开	备注		
_按钮1_鼠标位置被移动	逻辑型				
参数名	类型	参考	可空	数组	备注
横向位置	整数型				
纵向位置	整数型				
功能键状态	整数型				

标签1.可视 = 真

子程序名	返回值类型	公开	备注		
_启动窗口_鼠标位置被移动	逻辑型				
参数名	类型	参考	可空	数组	备注
横向位置	整数型				
纵向位置	整数型				
功能键状态	整数型				

标签1.可视 = 假

图 6-14 按钮提示信息代码示例

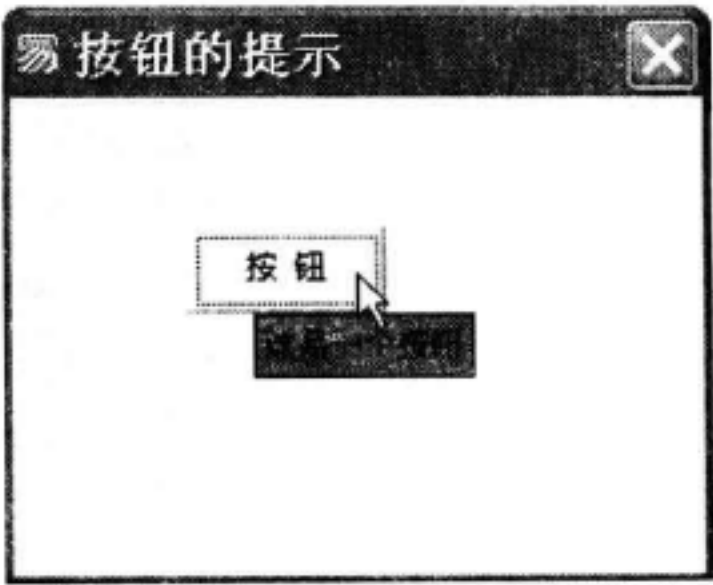


图 6-15 按钮提示信息运行结果

6.2 图形按钮

6.2.1 图形按钮概述

上一节介绍了按钮组件,其中用到了用图形来美化按钮。虽然效果美观,但是这样编程确很麻烦,因为每次都要将图片放到资源中去调用,即使使用外部图片也同样非常繁琐。这节就介绍一种简单的方法,可以一次放入所需的图片并直接使用,即使用图形按钮组件 \square 。该组件有4张不同的图片,分别显示不同的状态。

图形按钮的重要属性有“正常图片”、“点燃图片”、“按下图片”、“禁止图片”、“透明颜色”、“类型”、“选中”等。

图形按钮的重要事件有“被单击”。图形按钮没有专有方法。

6.2.2 图形按钮的属性

(1) “正常图片”、“点燃图片”、“按下图片”、“禁止图片”属性。

图形按钮的属性分别指定了不同状态下图片的样式:正常情况下,按钮上显示“正常图片”;鼠标在按钮上时,显示“点燃图片”;当按下鼠标时,显示“按下图片”;当按钮被禁止时,显示“禁止图片”。

新建一个易程序,在里面放一个图形按钮。在属性面板中用鼠标单击“正常图片”、“点燃图片”、“按下图片”、“禁止图片”属性,会弹出输入图片对话框。如图6-16所示。

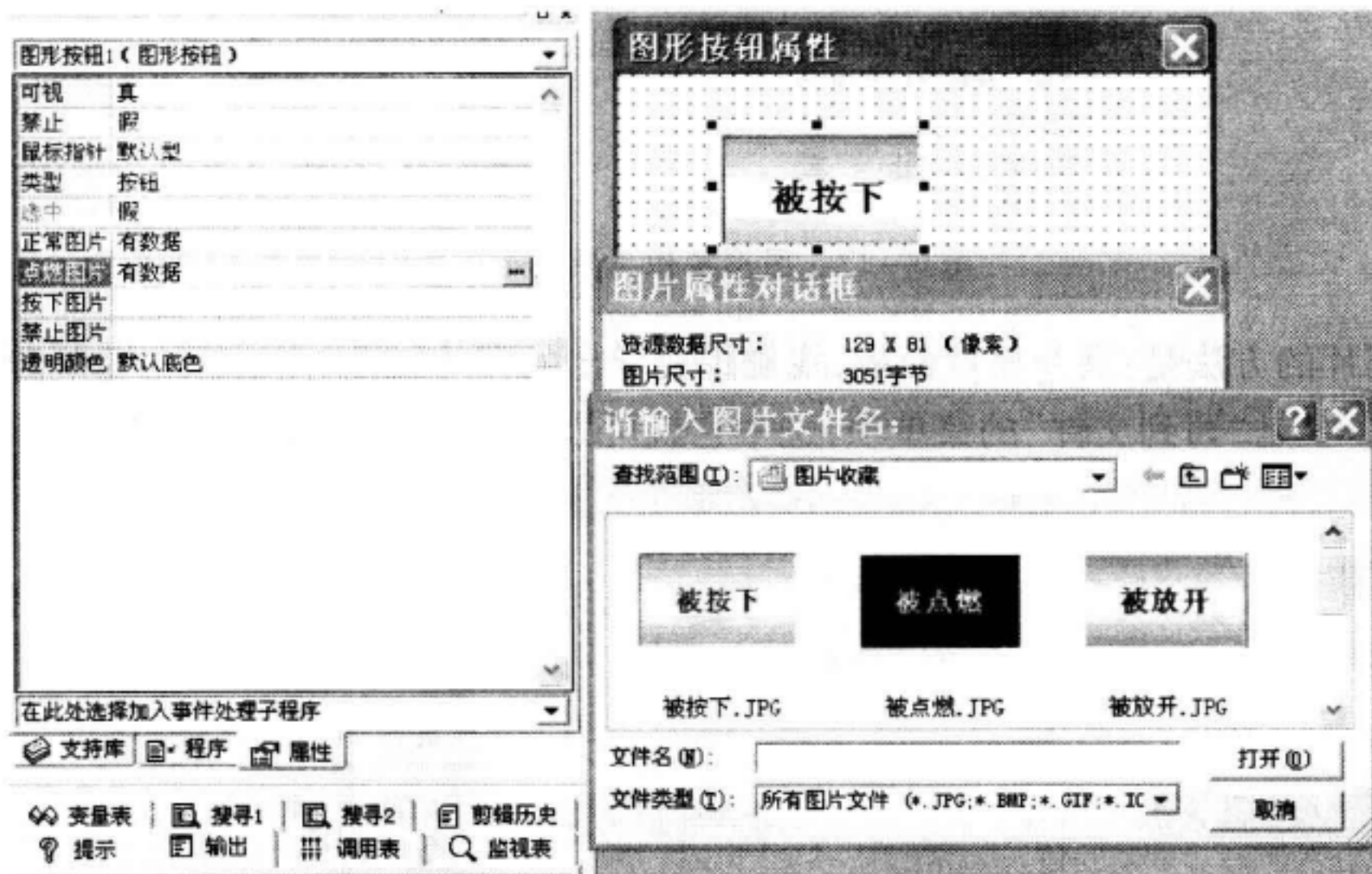


图 6-16 图形按钮属性对话框

【范例6-5】演示图形按钮组件的属性:“正常图片”、“点燃图片”、“按下图片”、“禁止图片”。正常情况下,按钮上显示“正常图片”;鼠标在按钮上时,显示“点燃图片”;当按下鼠标时,显示“按下图片”;当按钮被禁止时,显示“禁止图片”。具体参考例程6-5。程序代码如图6-17所示。

【运行结果】运行例程6-5,观测按钮的变化,如图6-18所示。

窗口程序集名	保留	保留	备注
窗口程序集1			

子程序名	返回值类型	公开	备注
_选择框1_鼠标左键被按下	逻辑型		
参数名	类型	参考	可空 数组 备注
横向位置	整数型		
纵向位置	整数型		
功能键状态	整数型		

```
如果 (选择框1.选中 = 真)
图形按钮1.禁止 = 假
图形按钮1.禁止 = 真
```


子程序名	返回值类型	公开	备注
_选择框1_按下某键	逻辑型		
参数名	类型	参考	可空 数组 备注
键代码	整数型		
功能键状态	整数型		

```
如果 (选择框1.选中 = 真)
图形按钮1.禁止 = 假
图形按钮1.禁止 = 真
```


子程序名	返回值类型	公开	备注
_选择框1_鼠标左键被放开	逻辑型		
参数名	类型	参考	可空 数组 备注
横向位置	整数型		
纵向位置	整数型		
功能键状态	整数型		

```
如果 (选择框1.选中 = 真)
图形按钮1.禁止 = 假
图形按钮1.禁止 = 真
```

图 6-17 图形按钮属性演示代码

导出图片的方法是:展开属性面板,找到想要导出的图片属性,用鼠标右键单击,会弹出菜单,其中有一个是“写到文件”的菜单项,选中它就可以导出图片了,如图 6-19 所示。

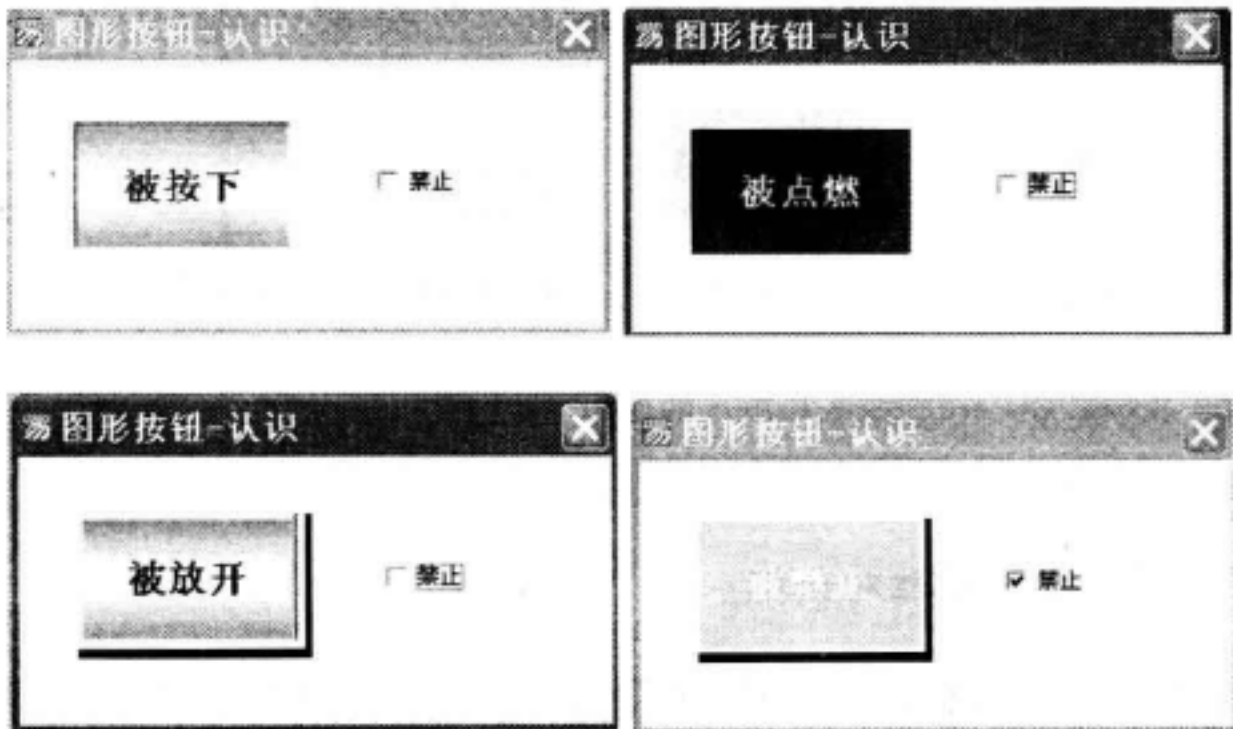


图 6-18 图形按钮属性运行结果

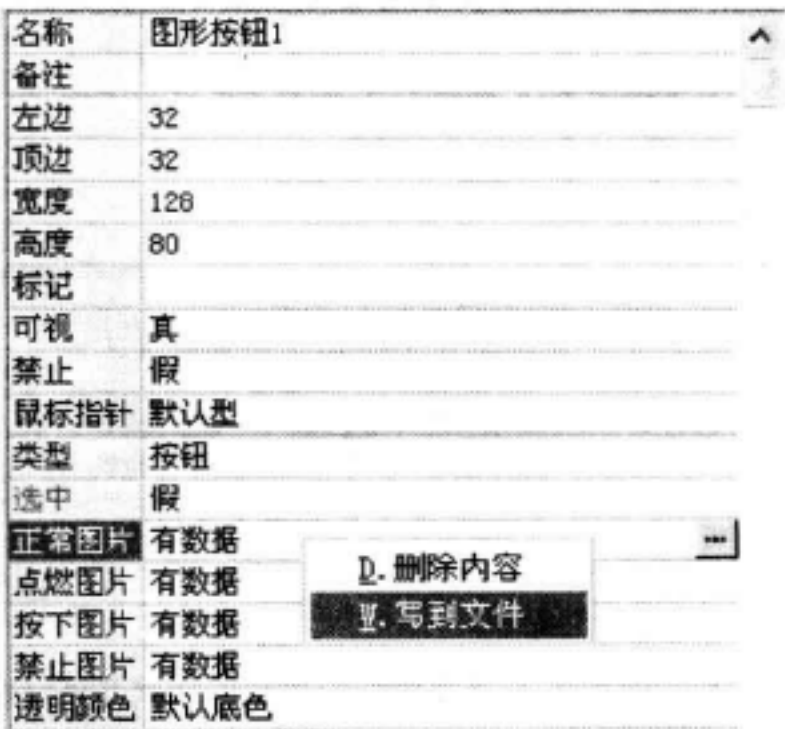


图 6-19 导出图片

删除图片的方法是:在属性面板中找到想要导出的图片属性,用鼠标右键单击,会弹出菜单,其中有一个是“删除内容”的菜单项,选中它,就可以删除图片了。不过最直接的作法是使

用 Del 键直接删除。

用程序实现删除图片的程序代码是：

图形按钮 1. 正常图片 = {}

(2) “透明颜色”属性属性可指定所有图片中透明部分的颜色。如果为“#默认底色”，则表示无透明色。

透明颜色请结合图 6-20 来理解。图中右侧的图片周围是白色的背景，看着觉得不舒服，但是把“透明颜色”设定为白色后，显示的结果就如左侧的样子了。可以看到周围的白色背景被滤掉了，这就是“透明颜色”属性的作用。图中左边是使用透明颜色为白时的效果，右边没有设置透明颜色。

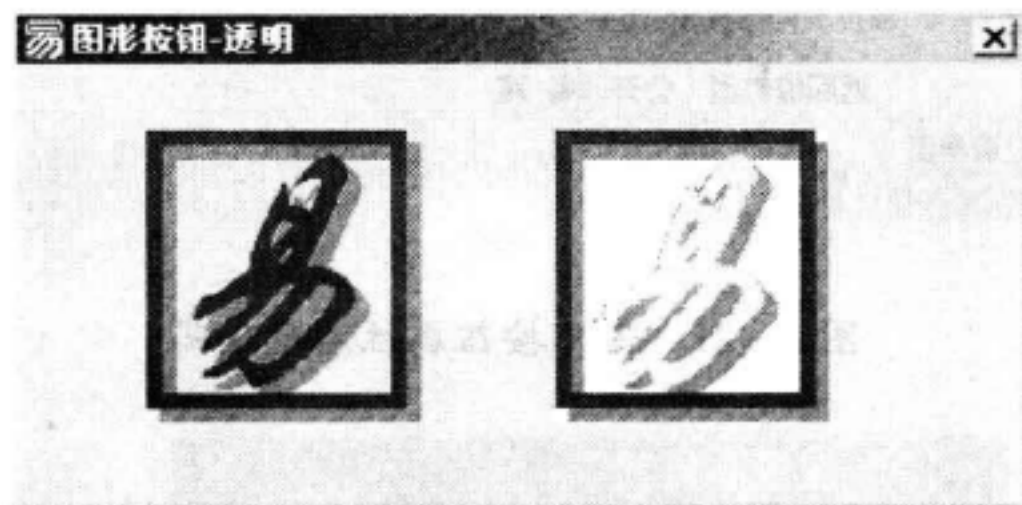


图 6-20 图片按钮透明属性

(3) “类型”、“选中”属性。

图形按钮的类型有两种：“0. 按钮、1. 选择框”，默认为“0. 按钮”。

一般情况下，设计程序使用最多的是“0. 按钮”类型。但“1. 选择框”类型也为设计程序时提供一个特别的思路，即用图形按钮组件代替选择框组件，这样就可以产生选择框组件的“选中”功能。

当“类型”为选择框(类型=1)时，“选中”属性开始有效，控制是否被选中，选中与否的两种外观由“正常图片”选中 = 假、“按下图片”选中 = 真来分别设置。

当类型=0 时(默认值)，图形按钮的“正常图片”、“点燃图片”属性中必须有图片；当类型=1 时，“正常图片”、“按下图片”属性中必须有图片。如果程序中不需要禁止图形按钮，可以不指定“禁止图片”属性。

6.2.3 图形按钮的事件

图形按钮组件与按钮组件一样，都有“被单击”、“被双击”、“鼠标左键被按下”、“鼠标左键被放开”、“鼠标右键被按下”、“鼠标右键被放开”、“鼠标位置被移动”7 个事件，仅比按钮组件少几个事件。与普通按钮组件的几个事件的使用上完全相同，在此不作介绍了。

图形按钮通常要和其他组件(如标签、图片框等)配合使用，这样才能构造出全新的界面出来。

【范例 6-6】演示图形按钮的“类型”、“选中”属性以及图形按钮的单击事件。当图形按钮被单击后，信息框提示显示“图形按钮被单击”，单击切换类型按钮，则实现图形按钮在选择框类型和按钮类型之间的互换。具体参考例程 6-6。

例程的具体代码如图 6-21 所示。

【运行结果】运行例程 6-6，观测分别单击 3 个按钮后的变化，如图 6-22 所示。

窗口程序集名	保留	保留	备注
窗口程序集1			

子程序名	返回值类型	公开	备注
_类型切换按钮_被单击			

图形按钮.禁止 = 假

如果 (图形按钮.类型 = 1)

 图形按钮.类型 = 0

 选中按钮.禁止 = 真

 图形按钮.类型 = 1

 选中按钮.禁止 = 假

信息框 (“当前类型为:” + 选择 (图形按钮.类型 = 0, “按钮”, “选择框”), 0,)

子程序名	返回值类型	公开	备注
_选中按钮_被单击			

图形按钮.选中 = 取反 (图形按钮.选中)

子程序名	返回值类型	公开	备注
_图形按钮_被单击			

信息框 (“图形按钮被单击”, 0,)

图 6-21 图形按钮属性演示代码

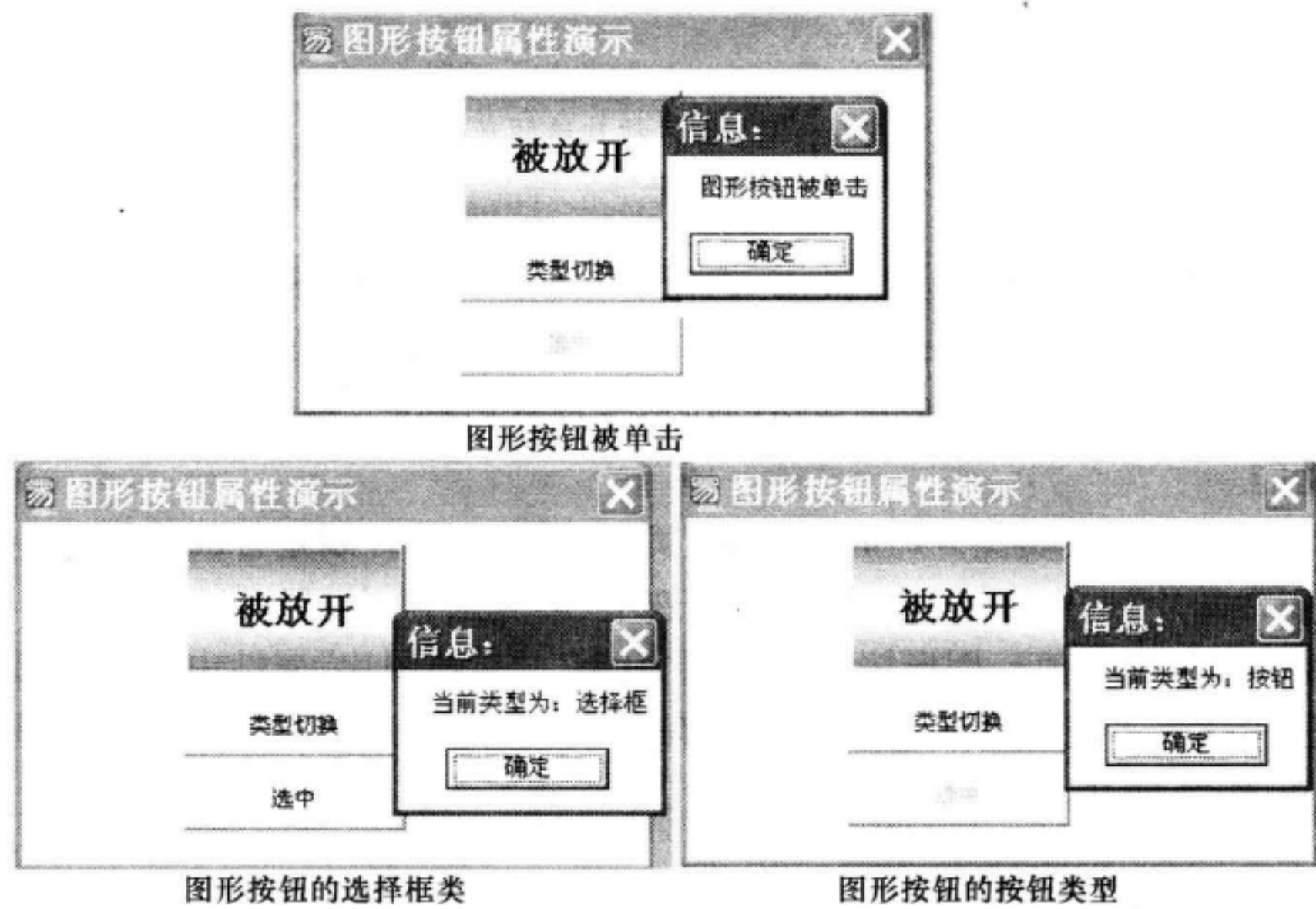



图 6-22 图形按钮类型运行结果

6.3 调节器组件

6.3.1 调节器组件概述

调节器组件  可以为图片框加上调节器,可以上下翻看图片;为标签加上调节器;在数据库应用中,可以为数据库翻看上一条下一条记录。

编辑框组件也有调节器,可以通过设置编辑框的“调节器方式”属性进行更改,本节中介绍二者的区别。

6.3.2 调节器的属性

1. “方向”属性

“方向”属性可控制调节器是横向还是纵向。本属性的取值为整数型,可为以下值之一:“0. 横向”、“1. 纵向”,默认为“1. 纵向”。该属性可以选中调节器后,在左边的属性窗口设置,如图 6-23、图 6-24 所示。

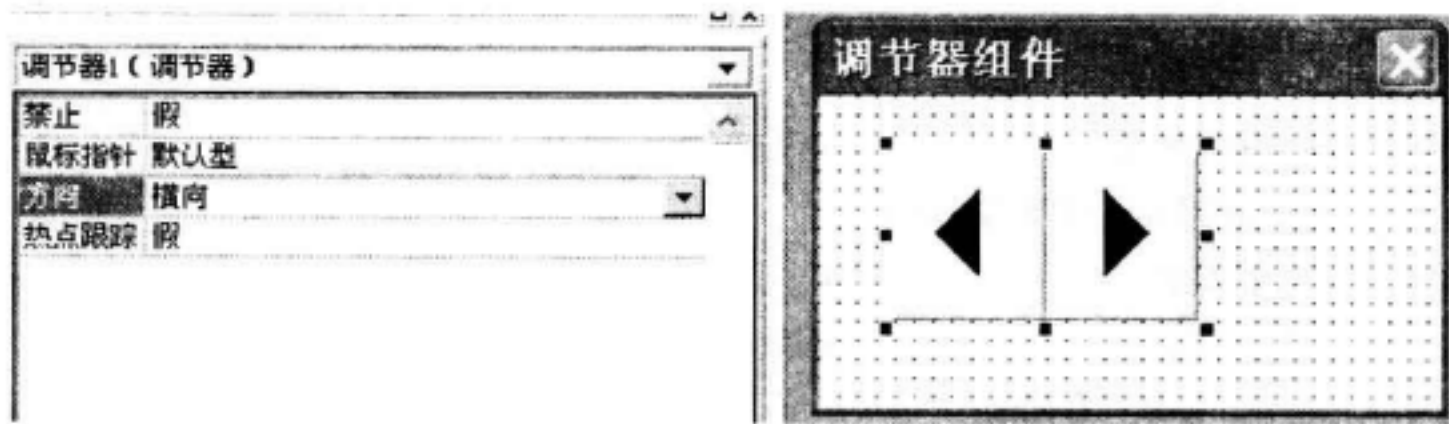


图 6-23 调节器的横向方向属性

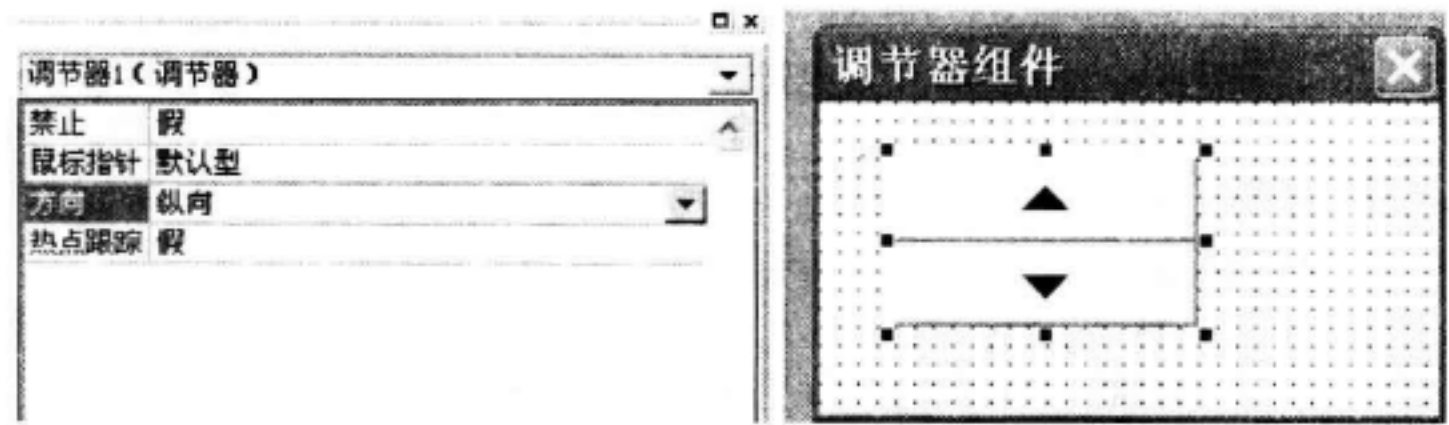


图 6-24 调节器的纵向方向属性

可以用代码设置其“方向”属性,例如:

```
调节器_方向. 方向 = 0
```

2. “热点跟踪”属性

“热点跟踪”属性是逻辑型属性,只能为“真”或“假”,默认为“假”。如果调节器的“热点跟踪”属性为“真”,则当鼠标指针移动到调节器的按钮上时,箭头的颜色会由黑色变为蓝色,为“假”时不变色,如图 6-25 所示。

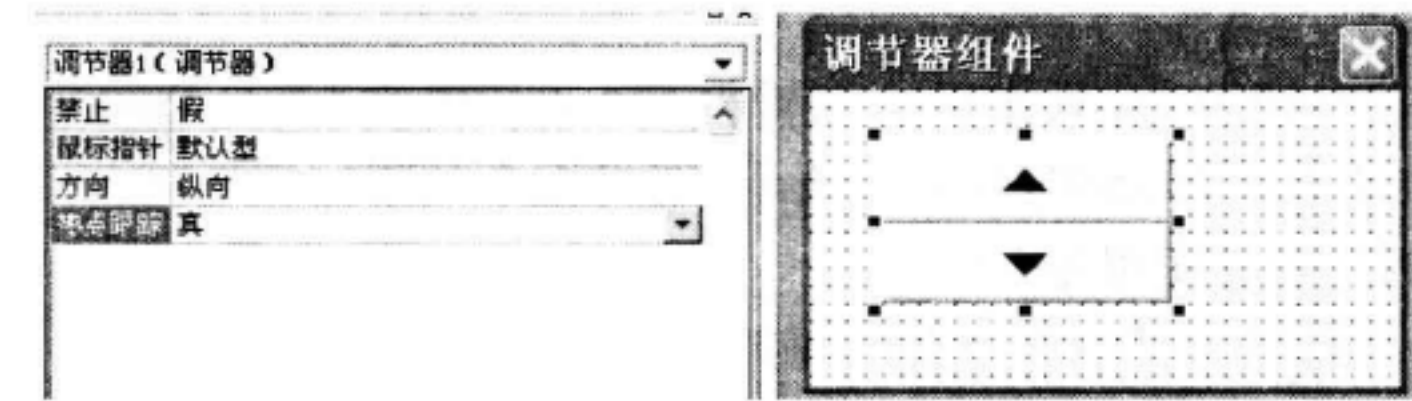


图 6-25 调节器的“热点跟踪”属性

用代码设置其“热点跟踪”属性,例如:

```
调节器_跟踪. 热点跟踪 = 真
```

6.3.3 调节器的重要事件

“调节钮被按下”事件:事件的产生时机是:当调节器组件的调节钮被按下时产生此事件。本事件有一个参数“按钮值”。如果按下的是调节器的向上/右箭头按钮,它的值为 1;如

果按下的是调节器的向下/左箭头按钮,它的值为-1。在本事件的处理子程序中可直接使用该参数。

【范例6-7】演示调节器按钮被按下事件。横向、纵向两个调节器的初始值均为0,若按下的调节器的向上/右箭头按钮,它的值为1;如果按下的是调节器的向下/左箭头按钮,它的值为-1,具体参考例程6-6。具体的程序代码如图6-26所示。

窗口程序集名	保留	保留	备注
窗口程序集1			

子程序名	返回值类型	公开	备注		
_调节器2_调节钮被按下					
参数名	类型	参考	可空	数组	备注
按钮值	整数型				

如果 (按钮值 = 1)

按下了向上的调节钮

编辑框1.内容 = 到文本 (到数值 (编辑框1.内容) + 1)

编辑框1.内容 = 到文本 (到数值 (编辑框1.内容) - 1)

子程序名	返回值类型	公开	备注		
_调节器1_调节钮被按下					
参数名	类型	参考	可空	数组	备注
按钮值	整数型				

如果 (按钮值 = 1)

按下了向上的调节钮

编辑框2.内容 = 到文本 (到数值 (编辑框2.内容) + 1)

编辑框2.内容 = 到文本 (到数值 (编辑框2.内容) - 1)

图 6-26 调节钮被按下代码

【运行结果】运行例程6-6,观测单击调节器的各个方向按钮后编辑框内数值的变化,如图6-27所示。

【注意】设置“编辑框”的“输入方式”属性为整数文本输入,确保只能输入数字而不能输入字母。

6.3.4 与编辑框调节器的对比

编辑框自己带有调节器,只需要在编辑框的“调节器方式”属性中选择自动调节器,如图6-28所示,编辑框中的手动调节器与调节器组件的使用是一样的。

在此,对两个组件进行了一些对比,见表6-1。

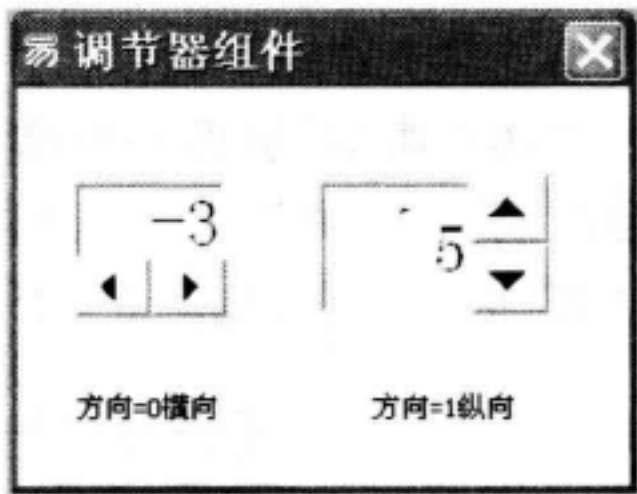


图 6-27 “调节钮被按下”
运行结果

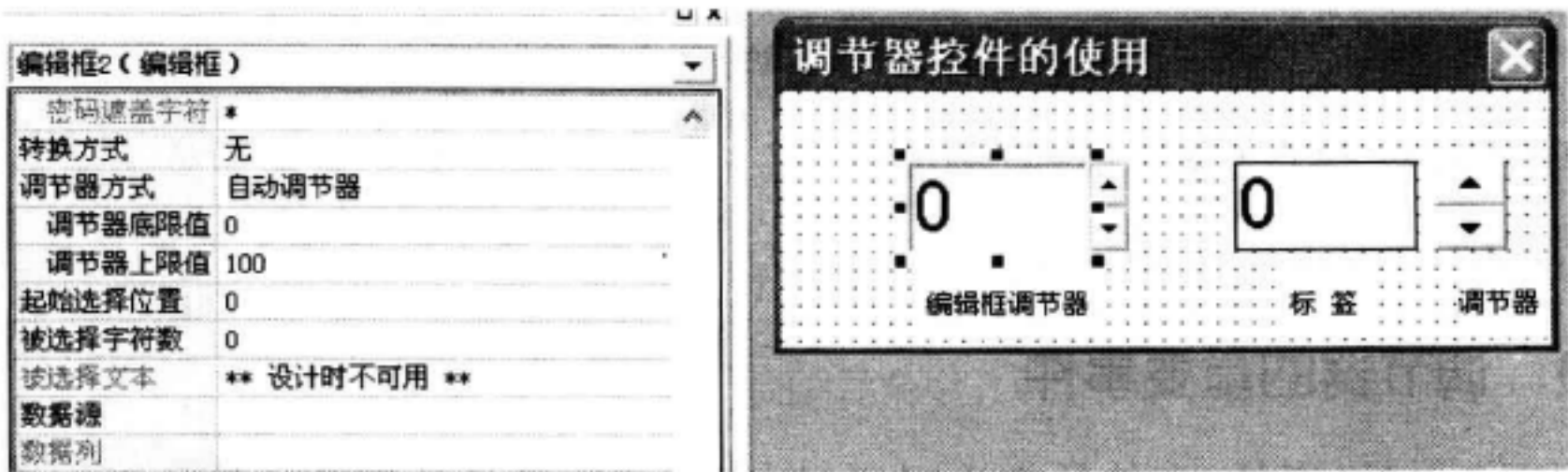


图 6-28 编辑框调节器与调节器

表 6-1 手动调节器与调节器组件对比

比较项目	编辑框组件的调节器	调节器组件
手动/自动	有	无,要程序实现
上限/下限	有	无,要程序控制
热点跟踪	无	有
方向	只有纵向	横向、纵向
大小可调	不可调,根据按钮大小	可任意大小

除了编辑框自带调节器,其他组件都没有调节器,因此调节器的用途非常广泛。如为图片框加上调节器,可以上下翻看图片。下面通过简单的例子认识其功能。

【范例 6-8】为图片框加上调节器,利用调节器上下翻看图片。具体参考例程 6-7,程序代码如图 6-29 所示。

【运行结果】运行例程 6-7,观测单击调节器的上/下按钮时图片框的变化,如图 6-30 所示。

窗口程序集名	保留	保留	备注
窗口程序集1			
变量名	类型	数组	备注
计数	整数型		

子程序名	返回值类型	公开	备注		
_调节器1_调节钮被按下					
参数名	类型	参考	可空	数组	备注
按钮值	整数型				

计数 = 按钮值 + 计数
如果真 (计数 % 3 = 1)
 图片框1.图片 = #图片2
如果真 (计数 % 3 = 2)
 图片框1.图片 = #图片3
如果真 (计数 % 3 = 0)
 图片框1.图片 = #图片1
标签1.标题 = 到文本 (计数)

子程序名	返回值类型	公开	备注
_启动窗口_创建完毕			

图片框1.图片 = #图片1
计数 = 0

图 6-29 调节器组件应用代码



图 6-30 调节器组件应用运行代码

6.4 拖放对象组件

6.4.1 拖放对象组件概述

拖放对象组件和其他通过鼠标点击激活事件的组件有些不同,该组件通过鼠标拖放触发事件,即鼠标按住一个目标向指定窗口拖动,就可以触发拖放对象组件的事件,从而取得相关信息。

6.4.2 拖放对象组件重要属性

拖放对象组件的重要属性有“接收文本”、“接收超文本”、“接收 URL”、“接收文件”,这些

属性都是逻辑型属性,用来设置拖放对象组件是否可以接收相关类型的信息,作用如下:

- “接收文本”属性设置拖放对象组件是否允许接收文本。
- “接收超文本”属性设置是否允许接收超文本。
- “接收 URL”属性设置是否允许接收 URL 地址。
- “接收文件”属性设置是否允许接收文件路径名。

当允许接收指定类型的内容后,同时该类型的内容被鼠标拖到指定的组件上时,则在鼠标指针下有一个小矩形和小加号。例如“接收文件”属性设置为“真”,拖放后效果如图 6-31 所示。

如果将“接收文件”属性设置为“假”,则不接收文件路径,拖放后的效果显示如图 6-32 所示。

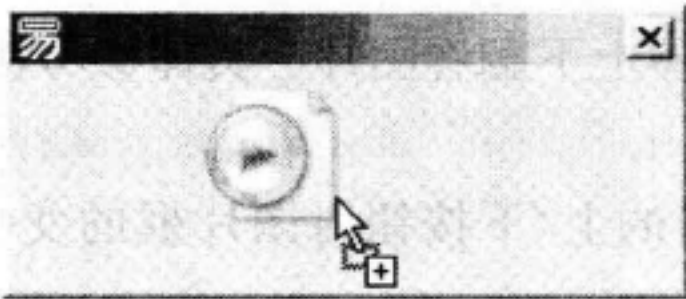


图 6-31 允许拖放文件的效果

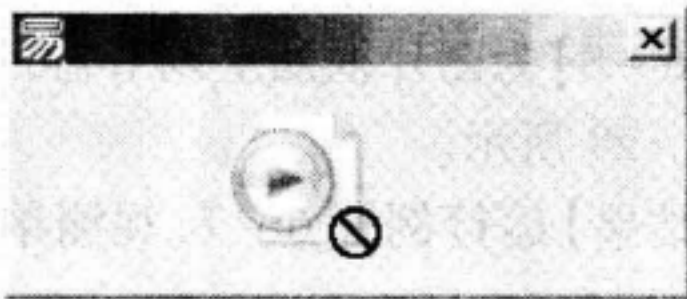


图 6-32 拒绝接收的效果

6.4.3 拖放对象组件的方法

拖放对象组件有 2 个方法:“注册拖放控件()”、“撤消拖放控件()”。拖放对象组件使用时,首先要运行“注册拖放控件()”方法,用来注册一个可以接收拖放内容的组件,在参数中填写欲注册组件的句柄(一个组件的句柄,是这个组件在内存中的位置,可以用“组件名.取窗口句柄()”方法取得)。

例如,让“拖放编辑框”接收拖放的内容,就将“拖放编辑框”的句柄填写到参数中,代码如图 6-33 所示。

子程序名	返回值类型	公开	备注
_允许拖放选择框_被单击			
如果 (允许拖放选择框.选中 = 真)			
拖放对象.注册拖放控件 (拖放编辑框.取窗口句柄 ())			

图 6-33 “注册拖放控件”方法

以上代码运行后,拖放编辑框就可以接收鼠标拖放进窗口中的内容了。接收到内容的拖放编辑框,就会触发鼠标拖放组件对应的事件。“撤消拖放控件()”方法,用来撤消已经注册过接收拖放的组件,在参数中填写欲撤消组件的句柄,例如撤消拖放编辑框的拖放功能,代码如下:

拖放对象 1. 撤消拖放控件 (_拖放编辑框.取窗口句柄 ())

6.4.4 拖放对象组件的事件

当注册了接收拖放内容的组件,并设置允许接收某类型的内容后,当被注册组件接收到拖放内容后,将触发拖放对象相应的事件,在时间的参数中,将提供接收到的内容。

1. “得到文本”事件

当注册组件接收到文本内容时,将触发本事件,并在本事件的参数中,返回接收到的文本内容。例如,将启动窗口接收到的文本,显示在编辑框中,代码如图 6-34 所示。

这里要注意,是否能接收到文本,也取决于组件的支持,例如系统自带的“记事本”程序,

子程序名	返回值类型	公开	备 注		
__拖放对象1_得到文本					
参数名	类 型	参 考	可 空	数 组	备 注
接收到的文本	文本型				

超级编辑框1.被选择文本 = 接收到的文本

图 6-34 “得到文本”事件代码

其中的编辑框就不能将选中文本拖拽出来;易语言提供的“超级编辑框”组件可以将选中的文本拖拽出来,还有很多组件也支持文本拖放。但网页中拖拽出来的选中文本,并不能触发“得到文本”事件,而是触发“得到超文本”事件,程序接收的是选中文本在网页中的源码。

2. “得到超文本”事件

当组件接收到网页中的文本内容时,触发本事件。当接收到网页中选中的文本时,本事件的参数返回该文本在网页中的源代码。具体的程序代码如图 6-35 所示。

3. “得到 URL”事件

当组件接收到网页中的超链接时,触发本事件。参数中返回接收到的 URL 地址。例如,使用超文本浏览框访问接收到的 URL 地址,具体的程序代码如图 6-36 所示。

子程序名	返回值类型	公开	备 注		
__拖放对象1_得到超文本					
参数名	类 型	参 考	可 空	数 组	备 注
接收到的超文本	文本型				

超级编辑框1.被选择文本 = 接收到的超文本

图 6-35 “得到超文本”事件

子程序名	返回值类型	公开	备 注		
__拖放对象1_得到URL					
参数名	类 型	参 考	可 空	数 组	备 注
接收到的URL	文本型				

超文本浏览框1.可视 = 真
超文本浏览框1.调整层次 0
__启动窗口_尺寸被改变 0
超文本浏览框1.地址 = 接收到的URL

图 6-36 “得到 URL”事件

4. “得到文件”事件

当接收到拖拽至组件上的文件时触发本事件,参数中返回接收到的文件路径。具体的程序代码如图 6-37 所示。

子程序名	返回值类型	公开	备 注		
__拖放对象1_得到文件					
参数名	类 型	参 考	可 空	数 组	备 注
接收到的文件路径	文本型				

变量名	类 型	静 态	数 组	备 注
找到位置	整数型			
取出长度	整数型			

超文本浏览框1.可视 = 假
__启动窗口_尺寸被改变 0
找到位置 = 寻找文本 (接收到的文件路径, #换行符, , 假)
如果 (找到位置 = -1)
取出长度 = 取文本长度 (接收到的文件路径)
取出长度 = 找到位置 - 1
超级编辑框1.内容 = 到文本 (读入文件 (取文本左边 (接收到的文件路径, 取出长度)))

图 6-37 “得到文件”事件

本节中引用的所有程序代码参见例程 6-9。

第 7 章 文本与图形组件

7.1 编辑框组件

7.1.1 编辑框概述

编辑框组件可以显示文本,也可以由用户编辑,是易语言中最基本、最常用的组件之一。虽然功能单一,但使用率是最高的。它的作用通常是提供程序使用者直接输入的区域。它在 Windows9X 系统下显示的文本容量很少,而在 WindowsNT/2000/XP 下的表现更好一些,使用的容量更大一些。

编辑框的主要属性有“名称、内容、起始选择位置、被选择字符数、被选择文本、是否允许多行、滚动条、调节器方式、转换方式、输入方式、最大允许长度、隐藏选择、文本颜色、背景颜色、字体、数据源、数据列”等。

编辑框的主要方法有“加入文本()”。

编辑框的主要事件有“内容被改变”、“调节钮被按下”。

7.1.2 编辑框的属性

1. “名称”、“备注”、“标记”属性

按钮组件有同等属性内容,此处略。

2. “内容”属性

“内容”属性即存放于编辑框中的文本。“内容”与“标题”属性的不同在于,“内容”属性可由程序使用者直接改动,而“标题”属性不可以。

编辑框组件只有“内容”属性,没有标题属性。“内容”是编辑框组件最重要的属性。

具体参见例程 7-1。观察一下,这个例子中只有一个编辑框,而有 3 个按钮在控制这个编辑框,如图 7-1 所示。

其中一个按钮可以改变编辑框中的内容。改变编辑框中的内容的程序代码如图 7-2 所示。

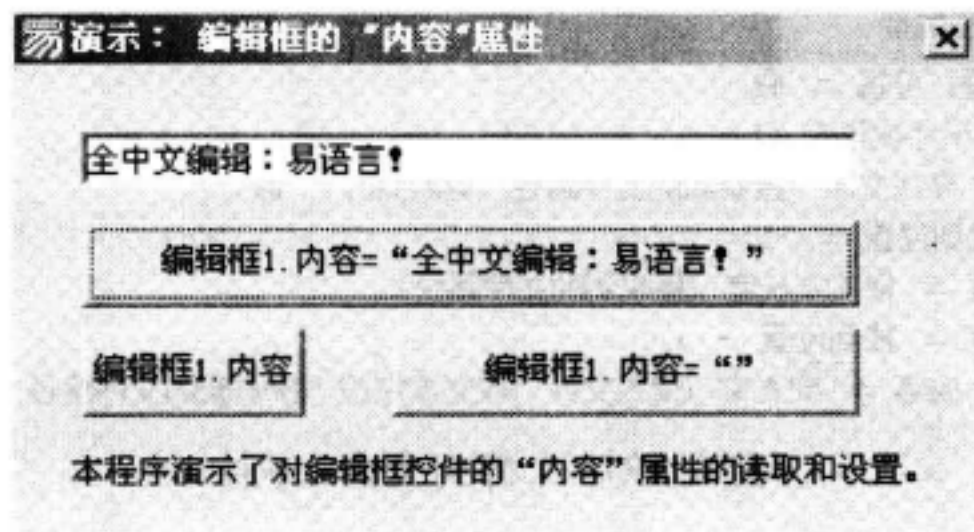


图 7-1 编辑框 - 内容属性运行结果

子程序名	返回值类型	公开	备注
_按钮1_被单击			
信息框(编辑框1.内容,0,)'把“编辑框1”的“内容”属性在信息框中显示出来。			
子程序名	返回值类型	公开	备注
_按钮2_被单击			
编辑框1.内容 = “全中文编辑:易语言!”			
子程序名	返回值类型	公开	备注
_按钮3_被单击			
编辑框1.内容 = “”			

图 7-2 程序代码

3. “起始选择位置”、“被选择字符数”、“被选择文本”属性

当编辑框中的文本有一部分处于“反选”状态时,“起始选择位置”属性描述了被选择文本的起始位置,(从0开始);如果编辑框中没有文本被选中,则“起始选择位置”指出光标所在位置(从0开始)。

“被选择字符数”属性描述了被选择文本的字符总数,汉字以两个字符计。

“被选择文本”属性包含了被选择的文本。

如果程序代码为“起始选择位置 = -1”,则将当前光标位置移动到文本尾部。如果为“被选择字符数 = -1”,则选择编辑框内的所有字符。如果在代码中为“被选择文本”赋值,则将编辑框中原来被选中的字符替换为新的字符串。

【注意】“被选择文本”属性只能在运行时设置,设计时不可改动。

默认情况下,“起始选择位置”、“被选择字符数”的值均为0,“被选择文本”的值为空文本(即“”)。具体参见例程7-2。本例程中,在设计时编辑框中没有任何文本,而由“__启动窗口_创建完毕”事件子程序在窗口建立时即将文本写入到编辑框中。如图7-3所示。

子程序名	返回值类型	公开	备注
_启动窗口_创建完毕			
编辑框1.加入文本 (“编辑框1,已设置:” + #换行符) 编辑框1.加入文本 (“是否允许多行=真” + #换行符) 编辑框1.加入文本 (“滚动条=2 (纵向滚动条)” + #换行符) 编辑框1.加入文本 (“隐藏选择=假 (失去焦点时仍可看到反选文本)” + #换行符 + #换行符) 编辑框1.加入文本 (“如果编辑框1中没有被选中的文本,则‘起始选择位置’返回当前光标的位置。” + #换行符 + #换行符) 编辑框1.加入文本 (“选择指定文本时要注意,是不能选择半个汉字的。” + #换行符) 编辑框1.加入文本 (“起始选择位置所在数据类型为编辑框,英文名称为SelStart,类型为整数型(int)。返回或设置所选择文本的起始点:0为位置1,1为位置2,如此类推。如果没有文本被选中,则指出光标位置。如果设置位置时使用值-1,则将当前光标位置移动到文本尾部。当类型为不可编辑下拉式时,本属性无效。”) 			

图 7-3 编辑框-选择类属性演示代码

该程序的运行结果如图7-4所示。

第1排第1个按钮主要针对于“起始选择位置”属性。运行时,任意用鼠标单击编辑框的某一位置后,再单击此按钮就会弹出信息框,以报告鼠标所在的第几个字符,如果是选中一些文字,单击这个按钮后,就会显示起始选中的位置。按钮被单击事件程序代码如下:

信息框(“起始选择位置:”+到文本(编辑框1.起始选择位置),0,)

第1排第2个按钮主要针对于“被选择字符数”属性,如果选中一些文字,单击这个按钮后,就会显示选择了多少个字符。如果什么也没有选,那么将显示为“0”。按钮被单击事件程序代码如下:

信息框(“被选择字符数:”+到文本(编辑框1.被选择字符数),0,)

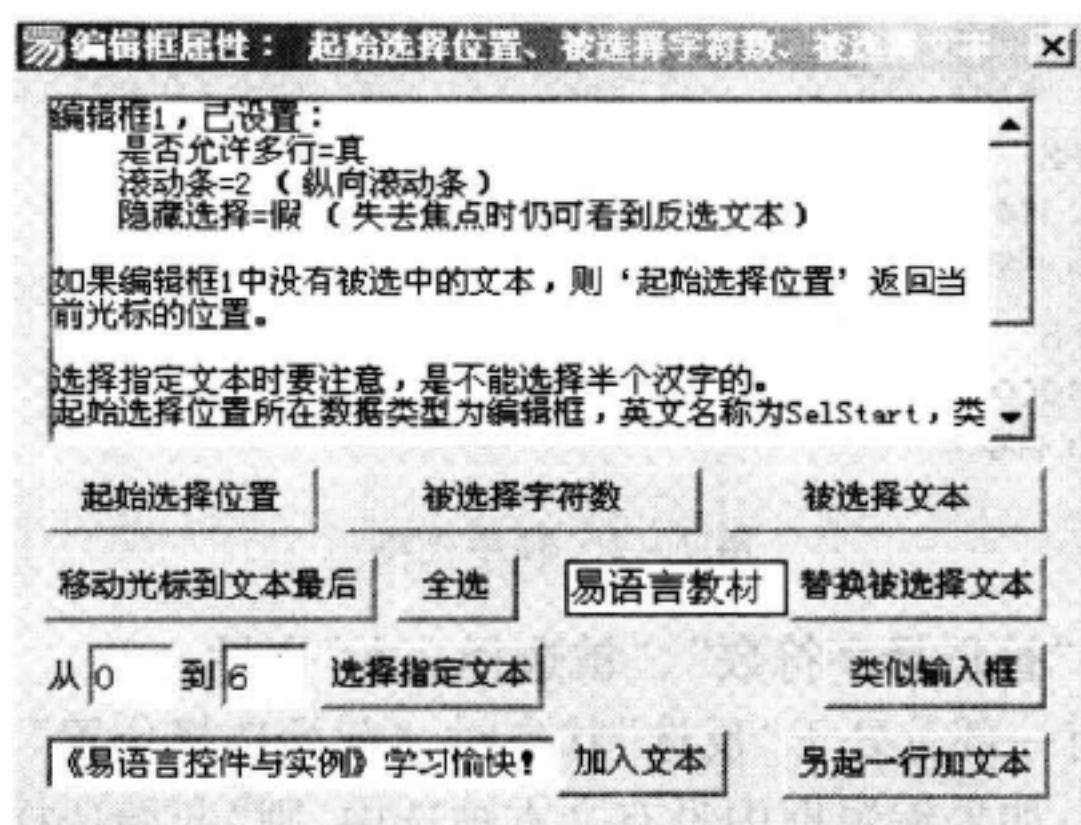


图 7-4 选择类属性运行结果

第 1 排第 3 个按钮主要针对于“被选择文本”属性,如果选中了一些文字,单击这个按钮后,就会显示所选择的文本。如果什么也没有选,那么将显示为空白。按钮被单击事件程序代码如下:

信息框(编辑框 1. 被选择文本,0,)

以上三个按钮事件,在主要代码后面还有一个“编辑框 1. 获取焦点()”的代码紧跟在后面,这是因为单击按钮时,按钮获得焦点,编辑框会同时失去焦点。这时恢复编辑框的焦点,否则看不到编辑框中的光标。

第 2 排第 1 个按钮即是对第 1 排第 1 个按钮的实际应用。

编辑框 1. 起始选择位置 = -1

“-1”值是易语言的属性所规定的。即:如果设置位置时使用值 -1,则将当前光标位置移动到文本尾部。大家可以在程序设计界面时的相关代码行上按 F1 键后就会显示即时帮助键,查看帮助。

第 2 排第 2 个按钮,可以全选编辑框中的文字。

编辑框 1. 被选择字符数 = -1

“-1”值是易语言的属性所规定的。即:如果设置字符数时使用值 -1,则选择文本框内的所有字符。大家可以在代码行上按 F1 即时帮助键,查看帮助。

第 2 排第 3 个按钮即是对第 1 排第 3 个按钮的实际应用。按钮被单击事件中代码如下:

编辑框 1. 被选择文本 = 编辑框 2. 内容

这行代码直接将编辑框 2 中的内容代替了编辑框 1 中被选择的文本,以达到替换被选择文本的目的。

下面讨论一下第 3 个按钮。

标题为“选择指定文本”的按钮被单击事件程序代码如下:

如果真(到数值(编辑框 4. 内容) > 到数值(编辑框 3. 内容))

编辑框 1. 起始选择位置 = 到数值(编辑框 3. 内容)

编辑框 1. 被选择字符数 = 到数值(编辑框 4. 内容) - 到数值(编辑框 3. 内容) + 1

如果真结束

代码中,第 1 行代码是对两个编辑框内输入的内容进行比较,左边的要比右边的要小一些,如果正常,将执行如果真判断语句内的代码,而此两行代码即可以根据“编辑框 3. 内容”指明“编辑框 1. 起始选择位置”,根据“编辑框 4. 内容”指明“编辑框 1. 被选择字符数”,两者合成,即好像是选中了一段编辑框 1 中的文字。

标题为“类似输入框”的按钮被单击事件程序代码如下:

局部变量:临时 数据类型:文本型

输入框(, , “请在这里输入”, 临时,)

这个按钮所产生的事件与上面的按钮是一样的,即选中的文本作为初始文本。

第 4 行的按钮中,标题为:“加入文本”的按钮被单击事件程序代码如下:

编辑框 1. 加入文本(编辑框 5. 内容)

此代码是使用了“加入文本()”的编辑框方法。将编辑框 5 中的内容加入到编辑框 1 内容的后面,是直接加入。

如果想要另起一行加入。请使用最后一个标题为:“另起一行加文本”的按钮,按钮被单击事件程序代码如下:

编辑框 1. 加入文本(#换行符 + 编辑框 5. 内容)

只不过是其中加入一个“#换行符 +”的命令,表示先换行再加入编辑框 5 中的内容。

具体参见例程 7-3,可以看到在“_启动窗口_创建完毕”事件子程序中有如下代码,运行结果如图 7-5 所示。

编辑框 1. 获取焦点()

编辑框 1. 内容 = “请重新输入口令”

编辑框 1. 起始选择位置 = 0

编辑框 1. 被选择字符数 = 14

可以看到,运行后,即将编辑框中的内容全部选中,如果这时输入新的口令,那么将会一下子将原编辑框中的内容去除,而立即显示输入的口令。

要注意第 1 行代码“编辑框 1. 获取焦点()”一定要有,如果没有此行代码,运行时光标将无法定位在编辑框中,也无法反选编辑框中的内容。再单击名称为:“类似输入框”的按钮,可以看到,这里是事先选中了初始文本,以提醒大家,如图 7-6 所示。

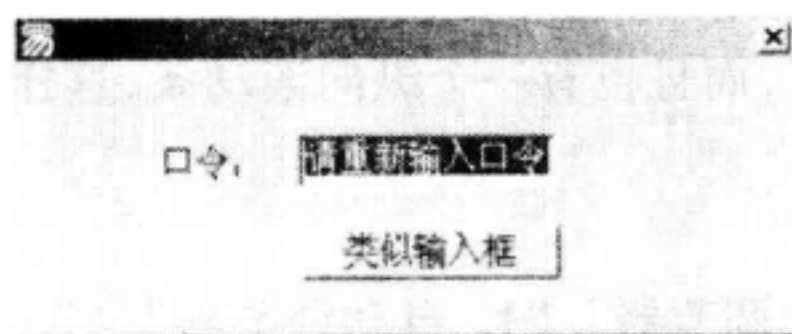


图 7-5 编辑框 - 类似输入框运行结果

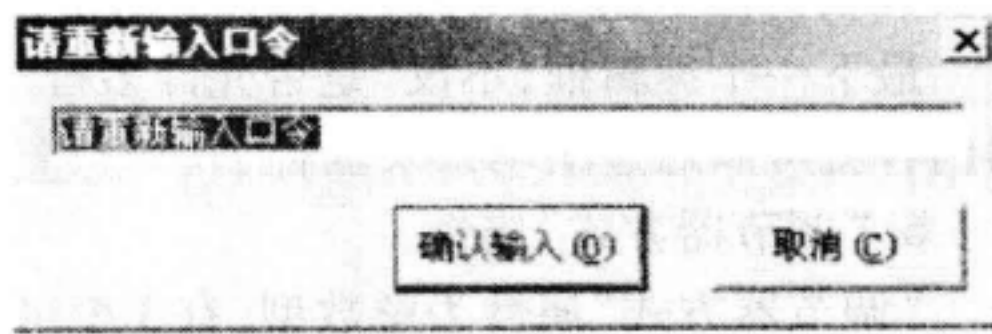


图 7-6 运行结果

使用到的程序代码如下:

子程序: _按钮_被单击

局部变量: 临时变量 数据类型: 文本型

输入框(, “请重新输入口令”, “请重新输入口令”, 临时变量,)

编辑框 1. 获取焦点()

4. “是否允许多行”属性

“是否允许多行”属性的值为逻辑型, 只能为“真”或“假”, 默认为“假”。当本属性为“假”时, 编辑框中只能输入一行文本。如果想要编辑框容纳多行文本, 就需要把该属性改为“真”。

编辑框. 是否允许多行 = 真

5. “滚动条”属性

整数型, 有以下4个可选值: “0. 无”、“1. 横向滚动条”、“2. 纵向滚动条”、“3. 横向及纵向滚动条”。默认为“0. 无”。只有当编辑框的“是否允许多行”属性为“真”时, 本属性才生效。

在多行编辑框中(即“是否允许多行”属性为真), 一般情况下把“滚动条”属性设为“2. 纵向滚动条”。

具体参见例程7-4。这个例程同时演示了“是否允许多行”与“滚动条”两个属性, 如图7-7所示。

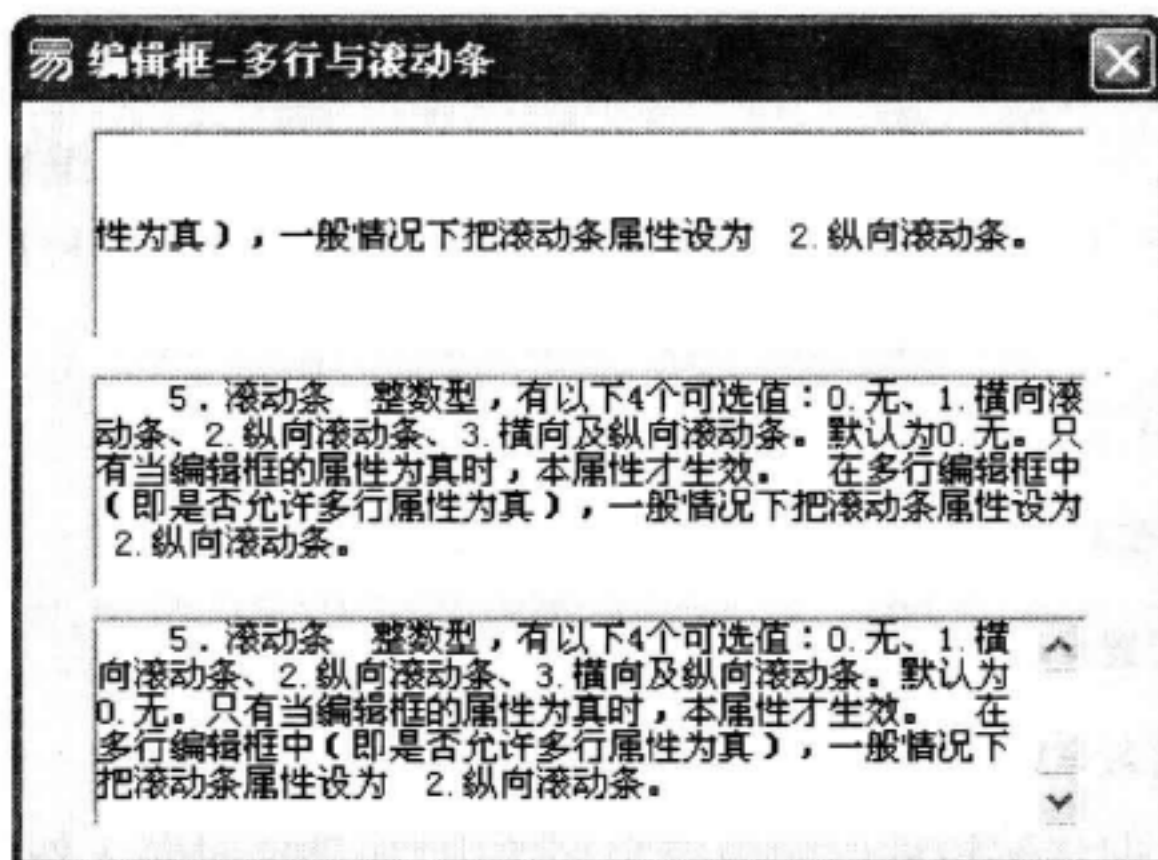


图 7-7 编辑框 - 多行与滚动条结果

其中最上一个编辑框即“是否允许多行”为“假”, 滚动条也没有。后面的内容都看不见。

中间一个编辑框“是否允许多行”属性为“真”, 但也没有滚动条, 如果要翻放更多的内容就要在编辑框中按下鼠标向下拉, 或用光标一格一格向下移, 非常不方便。

最下一个编辑框, 不仅“是否允许多行”属性为“真”, 而且也有一个纵向滚动条, 这样更多的内容就可以通过滚动条来翻看了。

6. “调节器方式”属性

“调节器方式”属性为整数型, 有3个可选值: “0. 无调节器”、“1. 自动调节器”、“2. 手动调节器”。默认值为“0. 无调节器”。本属性只有在单行方式(即“是否允许多行”属性为“假”)时才有效。

调节器用鼠标单击上下箭头, 即可自动变动编辑框中的数值。

如果需要调节器,通常设置为1,自动调节器,这样单击上下箭头,编辑框中的数字会自动加减1,而且还可以用另外两个属性“调节器底限值”(默认为0)、“调节器上限值”(默认为100)来控制数字变化范围。如果自动调节器不能满足要求,还可以采用手动调节器(置“调节器方式”属性为2),这时需要响应编辑框的“调节钮被按下”事件。

具体参见例程7-5,如图7-8所示。在这里,3个编辑框的调节器都设置为自动调节,因此,单击每一下调节器,增减数只为1。

在此例程中,在月份的内容被改变中加入一些代码,以判断大、小月问题,如图7-9所示。

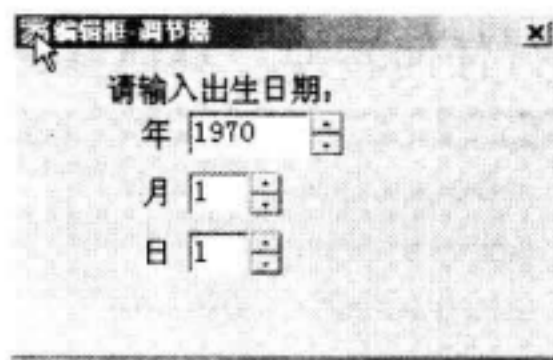


图7-8 编辑框-调节器
运行结果

窗口程序集名	保留	保留	备注
窗口程序集1			

子程序名	返回值类型	公开	备注
_编辑框2_内容被改变			

```

如果真 (删首尾空 (编辑框2.内容) = "1" 或 删首尾空 (编辑框2.内容) = "3" 或 删首尾空 (编辑框2.内容) = "5" 或 删首尾空 (编辑框2.内容) = "7" 或 删首尾空 (编辑框2.内容) = "9" 或 删首尾空 (编辑框2.内容) = "11" 或 删首尾空 (编辑框2.内容) = "13" 或 删首尾空 (编辑框2.内容) = "15" 或 删首尾空 (编辑框2.内容) = "17" 或 删首尾空 (编辑框2.内容) = "19" 或 删首尾空 (编辑框2.内容) = "21" 或 删首尾空 (编辑框2.内容) = "23" 或 删首尾空 (编辑框2.内容) = "25" 或 删首尾空 (编辑框2.内容) = "27" 或 删首尾空 (编辑框2.内容) = "29" 或 删首尾空 (编辑框2.内容) = "31")
编辑框3.调节器上限值 = 31

如果真 (删首尾空 (编辑框2.内容) = "4" 或 删首尾空 (编辑框2.内容) = "6" 或 删首尾空 (编辑框2.内容) = "8" 或 删首尾空 (编辑框2.内容) = "10" 或 删首尾空 (编辑框2.内容) = "12" 或 删首尾空 (编辑框2.内容) = "14" 或 删首尾空 (编辑框2.内容) = "16" 或 删首尾空 (编辑框2.内容) = "18" 或 删首尾空 (编辑框2.内容) = "20" 或 删首尾空 (编辑框2.内容) = "22" 或 删首尾空 (编辑框2.内容) = "24" 或 删首尾空 (编辑框2.内容) = "26" 或 删首尾空 (编辑框2.内容) = "28" 或 删首尾空 (编辑框2.内容) = "30")
编辑框3.调节器上限值 = 30

如果真 (删首尾空 (编辑框2.内容) = "2")
编辑框3.调节器上限值 = 28

```

图7-9 程序代码

【注意】本例程对于闰月只是简单判断了一下,并没有仔细判断,通过用时间日期类的命令进行判断,会更准确一些。

7. “输入方式”属性

“输入方式”属性为整数型,有以下可选值:“0. 通常方式”、“1. 只读方式”、“2. 密码输入”、“3. 整数文本输入”、“4. 小数文本输入”、“5. 输入字节”、“6. 输入短整数”、“7. 输入整数”、“8. 输入长整数”、“9. 输入小数”、“10. 输入双精度小数”、“11. 输入日期时间”。默认值为“0. 通常方式”。

当“输入方式”为1时(只读方式),程序使用者无法编辑编辑框中的内容。

当“输入方式”为2时(密码输入),输入编辑框中的字符将以“*”的形式显示(显示符号由编辑框的另一个属性——“密码遮盖字符”属性控制)。

当“输入方式”为3时(整数文本输入),只有数字字符(0,1,2,3...9)和负号(-)才可以输入到编辑框中,其余值类推。

具体参见例程7-6,该例程演示了所有输入方式运行后的结果。运行这个例程后,就可以任意输入内容,如图7-10所示。

8. “转换方式”属性

“转换方式”属性为整数型,有以下可选值:“0. 无”、“1. 大写->小写”、“2. 小写->大写”。默认为“0. 无”,不进行输入转换。

如果“转换方式=1”,则输入到编辑框中的字母全都显示为小写;如果“转换方式=2”,则输入到编辑框中的字母全都显示为大写。所有输入转换只涉及字母,非字母字符原样显示,不

进行任何转换。具体参见例程 7-7,运行结果见图 7-11。



图 7-10 编辑框-输入方式运行结果

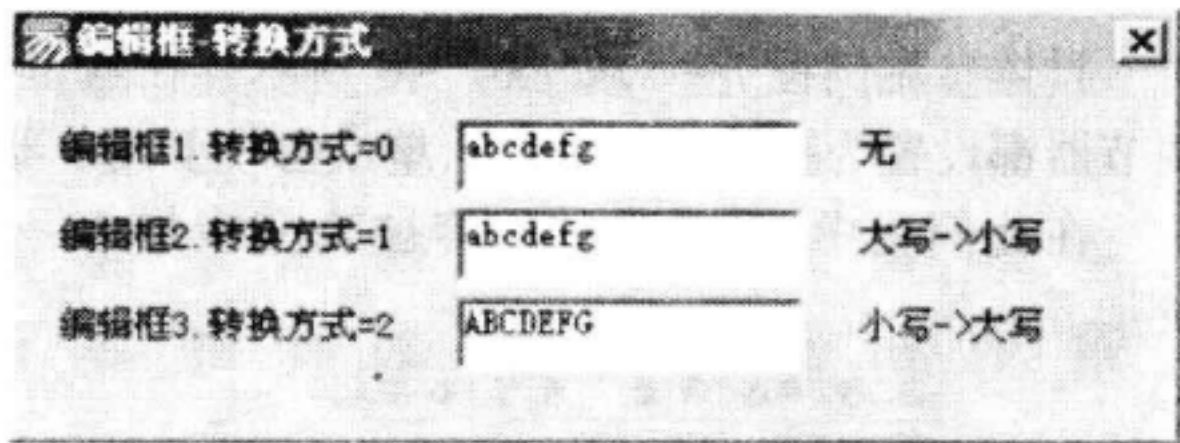


图 7-11 编辑框-转换方式运行结果

9. “最大允许长度”属性

“最大允许长度”属性为整数型,控制编辑框中所能容纳的最大字符数。默认为 0,表示不限制。

10. “隐藏选择”属性

逻辑型,只能为“真”或“假”,默认为“真”。如果本属性设置为“假”,则使编辑框失去输入焦点,原来反选的文字仍然反选。

11. “文本颜色”、“背景颜色”、“字体”属性

分别控制编辑框的文本颜色、背景颜色和字体,不再一一介绍。

12. “数据源”、“数据列”属性

“数据源”、“数据列”属性跟标签的同名属性类似,与数据库操作有关。

7.1.3 编辑框的方法

编辑框的方法由一个组件命令实现:“加入文本()”。可以通过观察支持库面板中“系统核心支持库”的“数据类型”看到这个方法,如图 7-12 所示。

“加入文本()”方法:

功能:在编辑框的最后(不管当前光标的位置)添加文本。

语法:编辑框名称.加入文本(欲加入文本,...)

参数:参数可以是字符串(如“易语言”),或文本型变量。最后一个参数可重复添加。

应用实例:

编辑框 1. 加入文本(“爱情”)

编辑框 1. 加入文本“爱情”,“如火”)

编辑框 1. 加入文本(“爱情”,“如火”,“燃烧”)

“加入文本()”只是在编辑框尾部(最后一个字符后面)添加指定文本,不会自动另起一

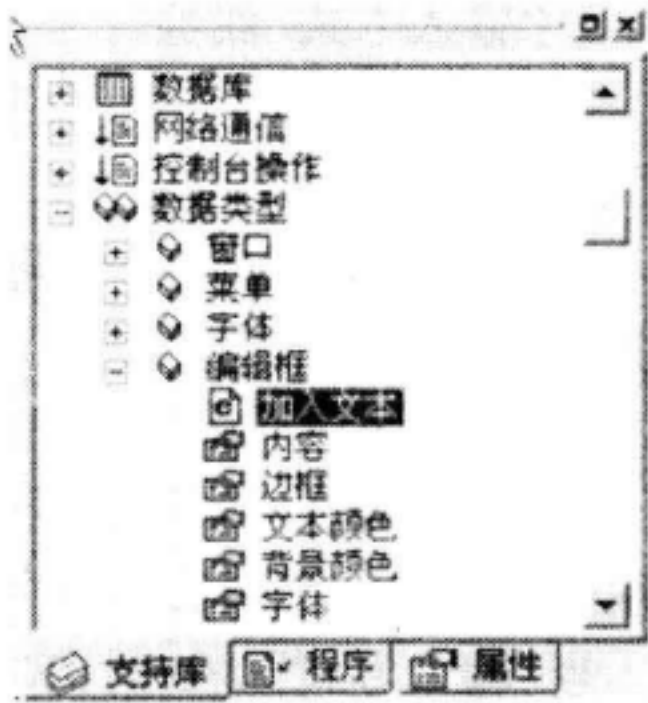


图 7-12 支持库面板中的编辑框方法

行。本“方法”在讲解编辑框的“起始选择位置”、“被选择字符数”、“被选择文本”属性时,提供的例程 7-2 中已经用到了,此处不再另外举例。

7.1.4 编辑框的事件

编辑框的重要事件有“内容被改变”、“调节钮被按下”。

1. “内容被改变”事件

事件的产生时机:当编辑框中的内容被改变时产生此事件。

用户向编辑框输入字符、或删除字符,或用代码改变编辑框的“内容”属性,都将引发“内容被改变”事件。

运行例程 7-5,可以看到,对于第 2 个代表月份的编辑框,使用了编辑框“内容被改变”的组件事件,这样,就通过判断大小月,以及闰月控制编辑框 3 显示的最大数为 31 天、30 天或 28 天。

参见例程 7-8,该例程演示了编辑框“内容被改变”的事件,运行结果如图 7-13 所示。

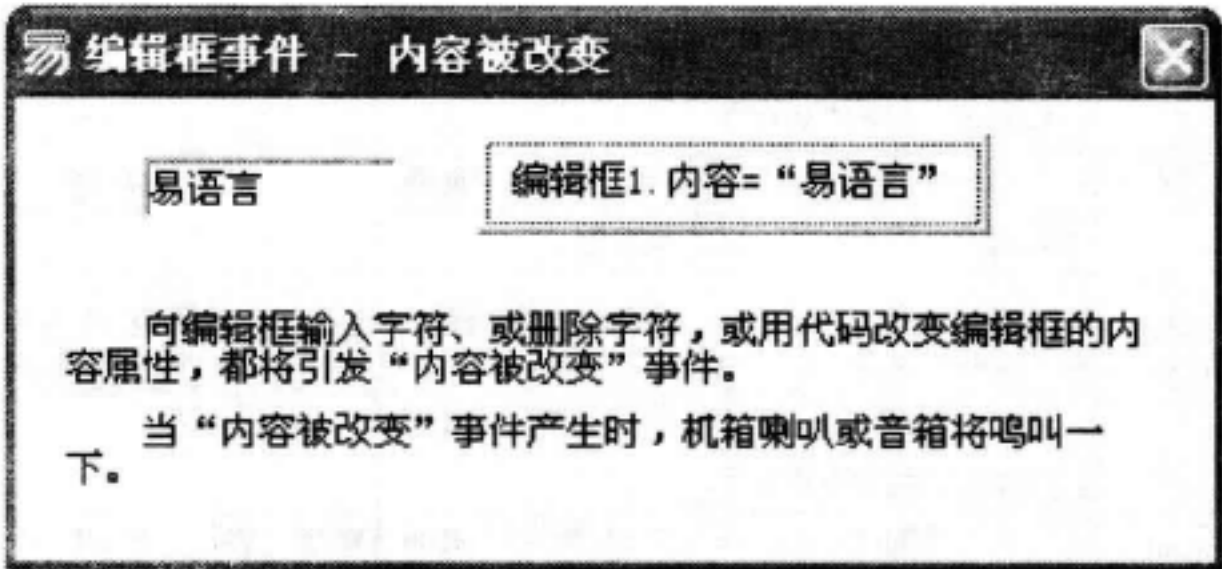


图 7-13 内容被改变事件运行结果

内容被改变子程序中有“鸣叫()”一行程序代码,如图 7-14 所示,所以编辑框 1 中只要改变内容就会鸣叫。

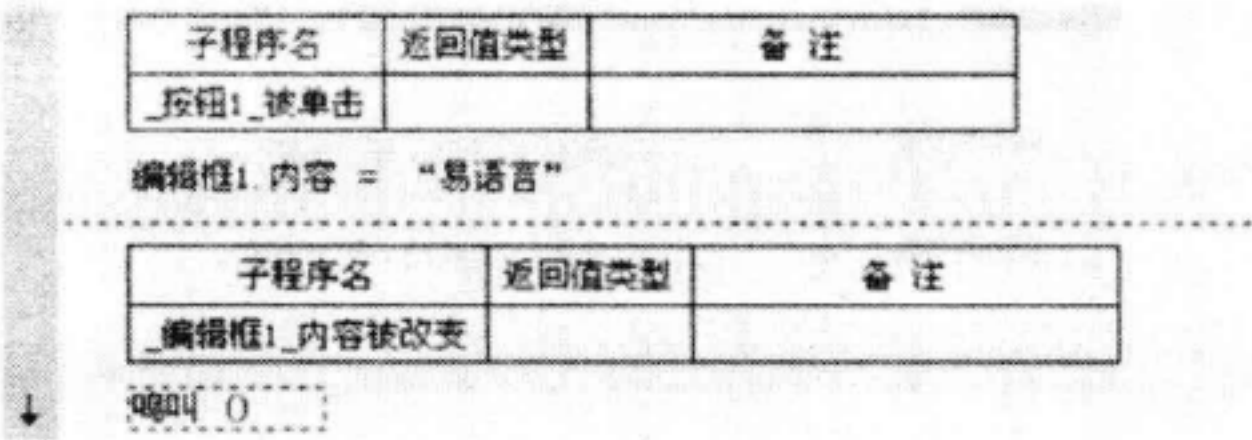


图 7-14 程序代码

2. “调节钮被按下”事件

事件的产生时机:当用户按下编辑框的调节钮时产生此事件。

本事件有一个整数型参数“按钮值”。如果用户按下的向上的箭头,按钮值 = 1;如果按下的是向下的箭头,按钮值 = -1。这个例程是介绍调节器方式时已经提供了的。

【注意】只有调节器方式 = 1(自动调节器)或 2(手动调节器)时,编辑框才会出现调节钮。而只有按下了“手动调节器”的调节钮时,“调节钮被按下”事件才会触发。

参见例程 7-9。运行该例程,分别单击不同的编辑框调节器,可以看到第 1 个编辑框每次只增减 1,而第 2 个编辑框每次增减 5,第 3 个编辑框每次增减 10,如图 7-15 所示。

3 个编辑框分别使用到的程序代码如图 7-16 所示。

注意,第一个编辑框不是自动调节器,是手动调节器,因此它的调节钮被按下事件中也要写入相关的代码,否则不起作用。

参见例程 7-10,该例程是自动调节器与手动调节器之间的比较,运行结果如图 7-17 所示,程序代码如图 7-18 所示。

3. “字符输入”事件

子程序名

返回值类型

公开

备注

_编辑框1_调节钮被按下

参数名

类型

参考

可空

数组

备注

按钮值

整数型

编辑框1.内容 = 到文本 (到数值 (编辑框1.内容) * 按钮值)

子程序名

返回值类型

公开

备注

_编辑框2_调节钮被按下

参数名

类型

参考

可空

数组

备注

按钮值

整数型

编辑框2.内容 = 到文本 (到数值 (编辑框2.内容) * 按钮值 * 5)

子程序名

返回值类型

公开

备注

_编辑框3_调节钮被按下

参数名

类型

参考

可空

数组

备注

按钮值

整数型

编辑框3.内容 = 到文本 (到数值 (编辑框3.内容) * 按钮值 * 10)

图 7-16 程序代码

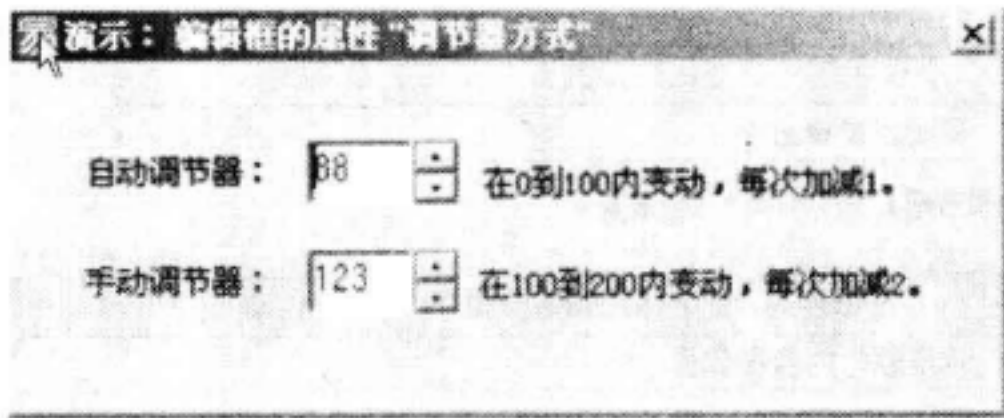


图 7-17 编辑框-调节器比较运行结果

“字符输入”事件的产生时机:用户按下键盘上的可打印字符键(包括回车、退格键)时。
语法:_组件名称_字符输入(字符代码)
参数:“字符代码”——被按下的字符键的 ASCII 码,均已被定义为常量,见常量表。
“字符输入”事件与“按下某键”事件、“放开某键”事件的区别在于:前者只接受可打印字符键;而后两者可接受所有键盘输入。
参见例程 7-11,运行后该例程后,在编辑框中输入“abcd”,会看到结果有不同的改变,如图 7-19 所示。
在图 7-19 下排的左边输入框中打回车键,喇叭或音箱会叫一声,但是右边的却不会。程序代码如图 7-20 所示。

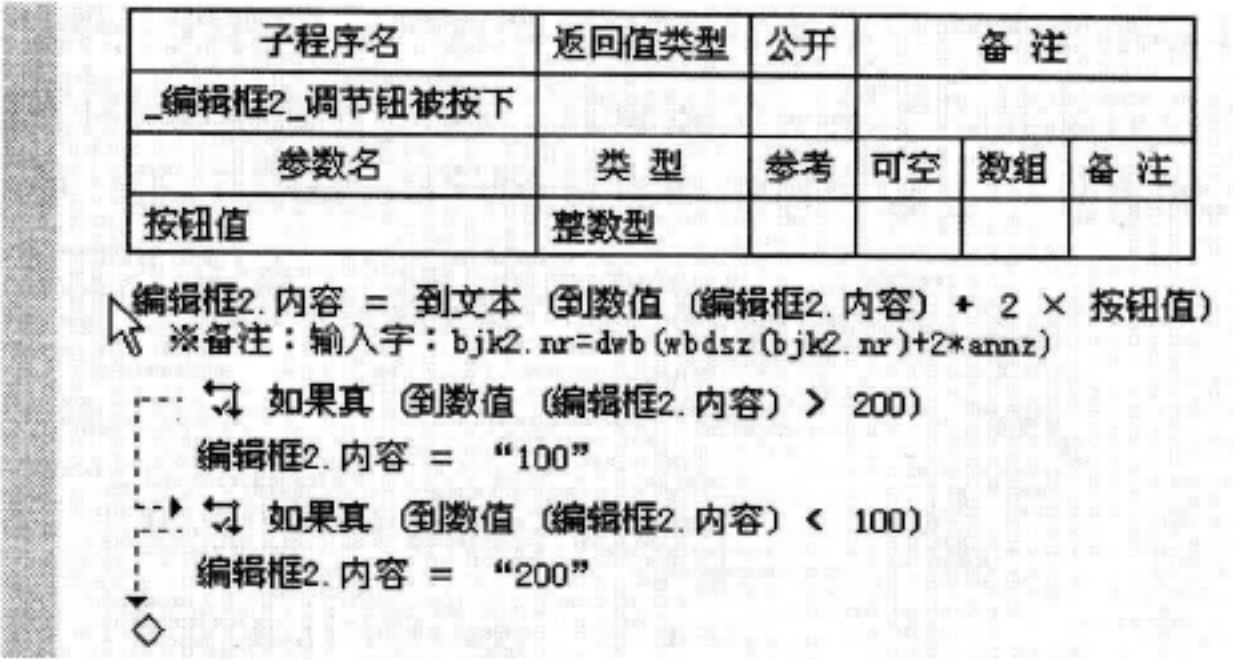


图 7-18 程序代码

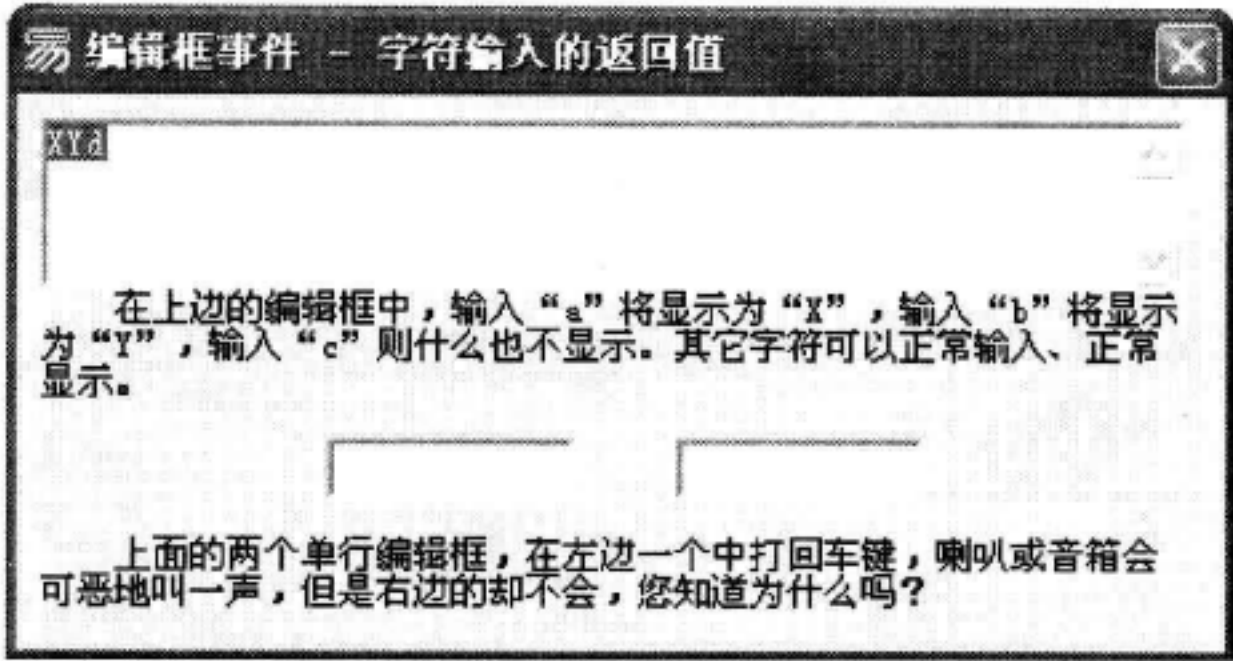


图 7-19 运行程序

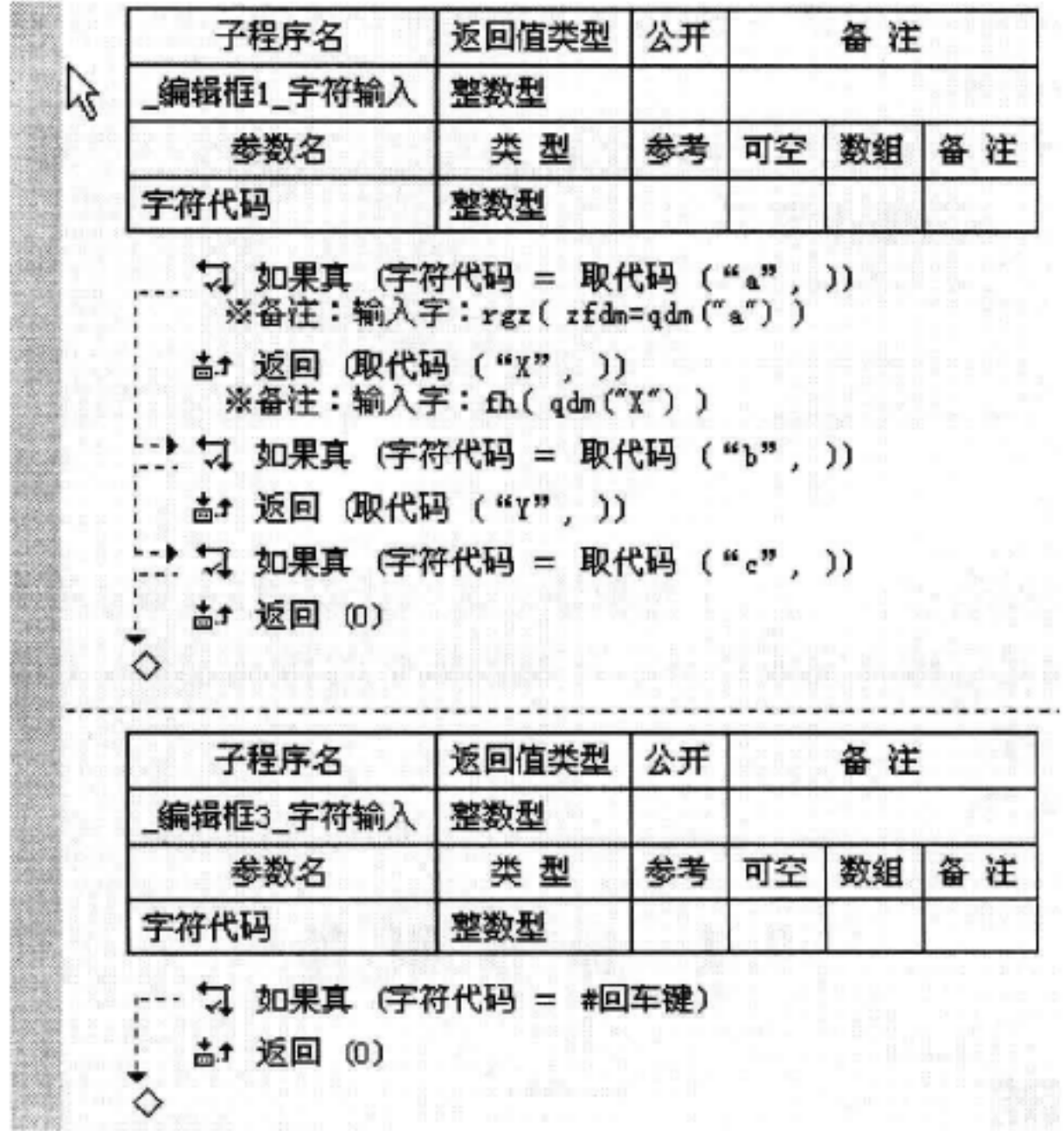


图 7-20 程序代码

原来都是通过字符输入事件子程序,一个实现了字母替换,另一个是实现鸣叫的功能。现在可以测试一下:内容被改变事件与字符输入事件到底哪一个更快一些。

新建一个易程序,在启动窗口上放一个编辑框。在内容被改变事件中加入弹出信息框的程序代码,在字符输入的事件中也加入弹出信息框的代码,如图 7-21 所示。

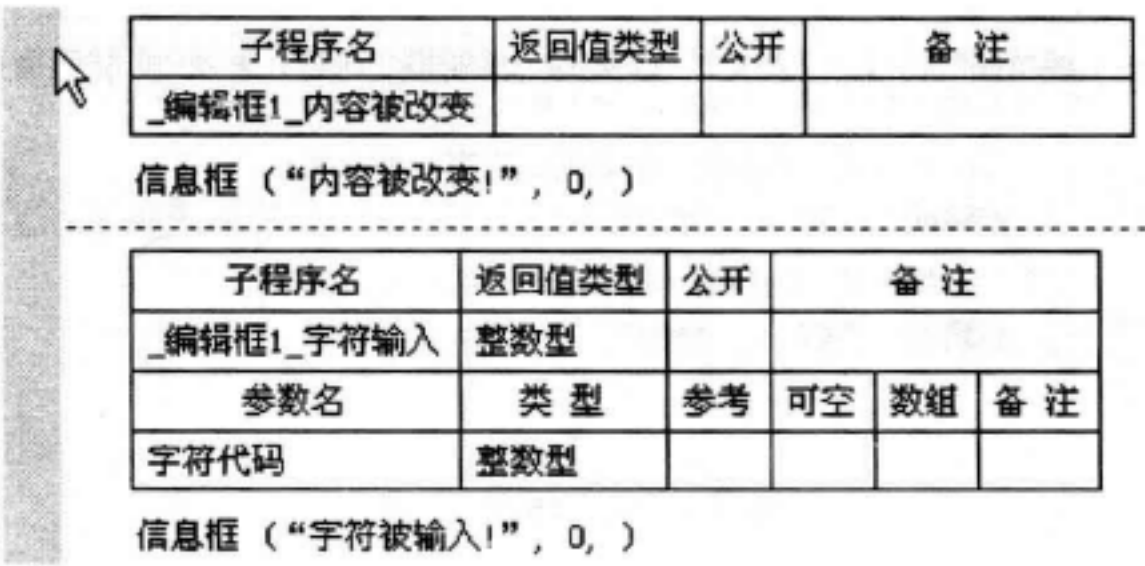


图 7-21 程序代码

现在试运行这个程序,在编辑框中输入任意字母,可以看到在字母还没有显示之前,字符被输入事件触发了,如图 7-22 所示。

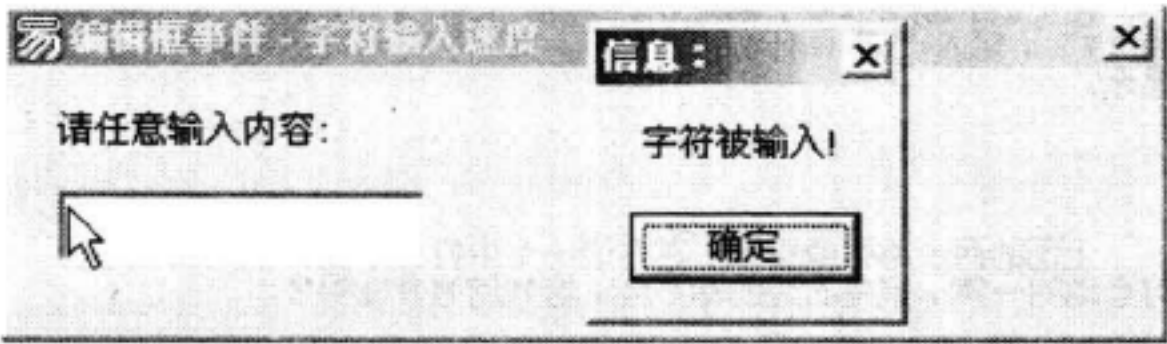


图 7-22 运行程序

当单击“确定”按钮后,就会继续运行,这时编辑显示了输入的字母,最后触发了内容被改变的事件,如图 7-23 所示。

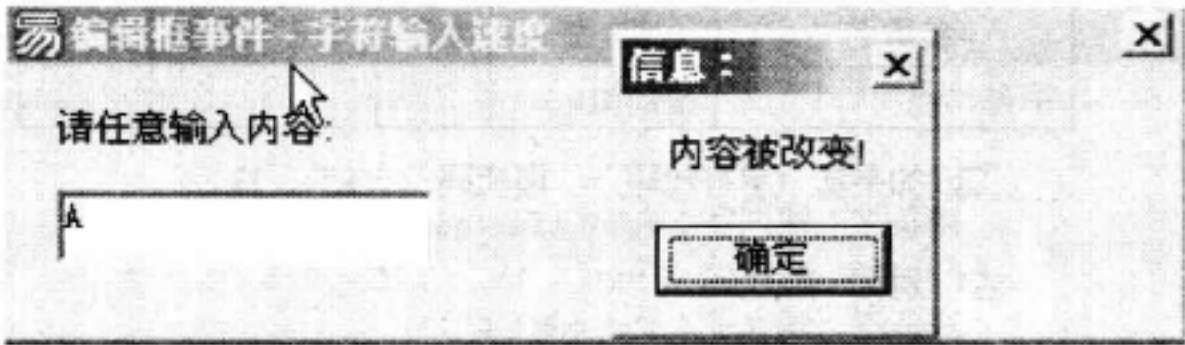


图 7-23 字符输入事件运行结果

通过这个例程的测试,可以看到,字符被输入是在按键盘后,显示在编辑框之前所作的拦截动作,而内容被改变是已输入完成并显示在编辑框中的检查动作。因此两者是有非常大的区别的,作用也不太一样。

具体参见例程 7-12。

7.2 标签组件

7.2.1 标签概述

标签也是使用频率非常高的组件。它的作用主要是显示某些提示性文字,还可以用来美化程序界面,最重要的一点是,它也可当作按钮组件来使用,这在按钮组件章节中已经作了一

些介绍。因此只要某组件有被鼠标单击的事件,就可用标签来模拟按钮组件。

标签的重要属性有“名称、标题、效果、边框、文本颜色、背景颜色、字体、底图、渐变背景方式、横向对齐方式、纵向对齐方式、是否自动折行、数据源、数据列”。

标签没有重要方法。标签有基本的 7 种事件。

7.2.2 标签属性

1. “名称”、“标题”属性

“名称”、“标题”属性的使用方法基本与按钮组件一样,在此略。

参见例程 7-13,并试运行该例程,其中“标题”属性可以随着按钮的单击而改变,如图 7-24 所示。

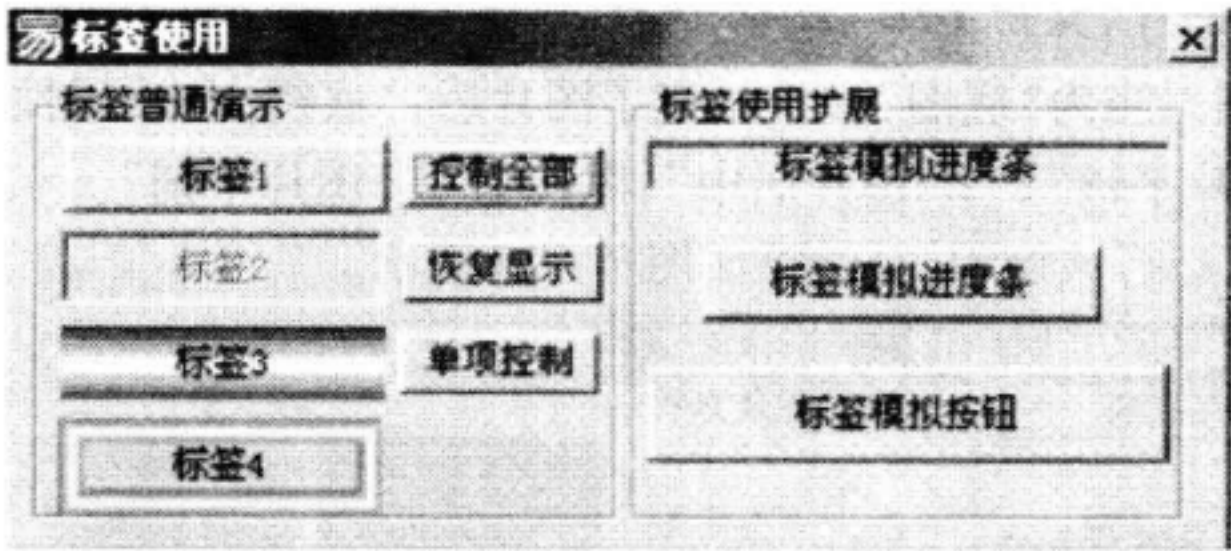


图 7-24 标签-使用运行结果

2. “效果”属性

有 4 个可选值:“0. 通常”、“1. 凹入”、“2. 凸出”、“3. 阴影”。描述标题文字,其外观如图 7-25 所示。

3. “边框”属性

“边框”属性有 7 个可选值:“0. 无边框”、“1. 凹入式”、“2. 凸出式”、“3. 浅凹入式”、“4. 镜框式”、“5. 单线边框式”、“6. 渐变镜框式”,默认为“0. 无边框”。

当“边框”的值为 6,即渐变镜框式时,以下几个属性将起作用:“渐变边框宽度”、“渐变边框颜色 1”、“渐变边框颜色 2”、“渐变边框颜色 3”,分别指定渐变边框的 3 个颜色。其中渐变边框宽度默认为 8,最小允许值为 5。渐变边框颜色 1、2、3 分别指定渐变中的 3 种颜色。

参见例程 7-14,运行结果如图 7-25 所示。可以看到不同的边框属性改变后的不同样子,以及效果、字体、颜色等。

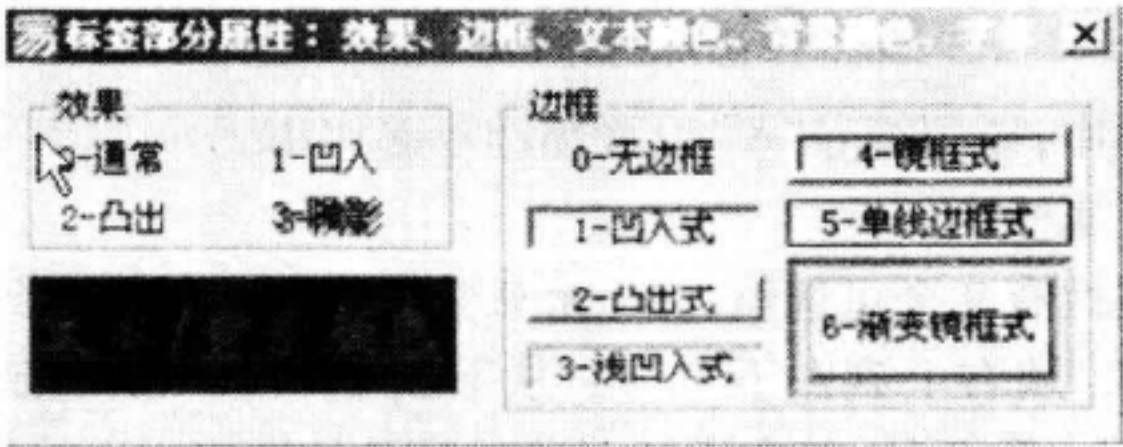


图 7-25 标签属性运行结果

本例程分别演示标签的“效果”、“边框”、“文本颜色”、“背景颜色”属性。

4. “文本颜色”、“背景颜色”属性

分别指定标签上的文本的颜色,指定标签背景的颜色,如图 7-25 所示。

标签. 文本颜色 = #红色

5. “字体”属性


“字体”属性指定标签上文本的字体。对字体颜色的设置无效, 文本颜色由“文本颜色”属性指定。

程序设计中单击“字体”属性栏, 可以直接为标签的字体设置属性。而用程序代码设置不同的字体与编辑框设置字体的一样。例程如下:

标签 1. 字体. 字体大小 = 24

标签 1. 字体. 字体名称 = “黑体”

6. “底图”、“底图方式”属性

指定标签的背景图片, 支持 BMP、JPEG、GIF、ICO、CUR 等格式。可在“底图”属性栏单击  按钮为标签加入底图。当为“底图”属性指定了图片后, “底图方式”属性生效, 可选项为: “0. 图片居左上”、“1. 图片平铺”、“2. 图片居中”, 默认为“1. 图片平铺”。

参见例程 7-15, 运行该例程, 其结果如图 7-26 所示。



图 7-26 底图方式、渐变背景方式运行结果

【注意】当设置“底图方式”属性为“2. 图片居中”时, 如果图片的实际尺寸大于标签的尺寸, 图片会自动缩放, 如果图片小于标签则原样显示。

另外, 既然标签可以设计底图, 而设置底图后上面的文字还存在, 并且标签具有“鼠标左键被按下”等事件, 因此, 比使用按钮组件设置图片更有优越性, 也可以看作是按钮组件的另一种形态。

【注意】在标签组件中, 使用符号“&”设置热键不起作用。

7. “渐变背景方式”属性

“渐变背景方式”属性有 9 个可选值: “0. 无渐变背景”、“1. 从上到下”、“2. 从左到右”、“3. 从左上到右下”、“4. 从右上到左下”、“5. 从下到上”、“6. 从右到左”、“7. 从右下到左上”、“8. 从左下到右上”。除 0 外, 其余描述了背景颜色的渐变方向。默认值是“0. 无渐变背景”。

当“渐变背景方式”为非 0 值时, 以下几个属性将起作用: “渐变背景颜色 1”、“渐变背景颜色 2”、“渐变背景颜色 3”, 分别指定渐变背景的 3 个颜色。

8. “横向对齐方式”、“纵向对齐方式”属性

分别描述标签上的标题文字在标签中的位置。“横向对齐方式”属性有3个可选值：“0. 左对齐”、“1. 居中”、“2. 右对齐”。“纵向对齐方式”属性也有3个可选值：“0. 顶对齐”、“1. 居中”、“2. 底对齐”。这两个属性的默认值都是“1. 居中”。

具体参见例程7-16,运行结果如图7-27所示。

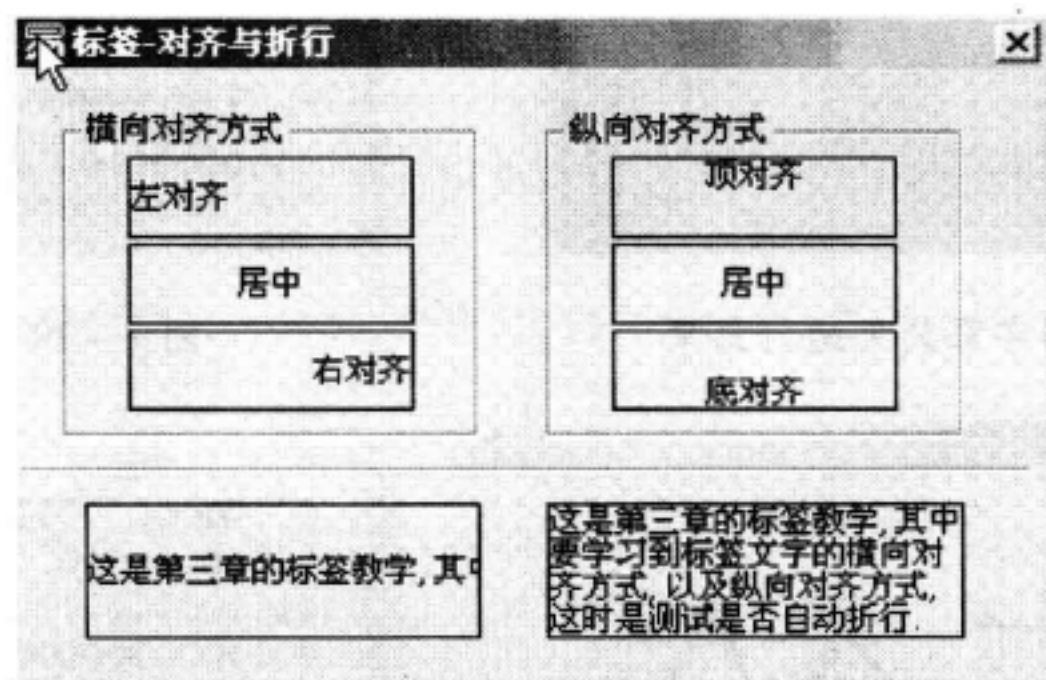


图 7-27 对齐与折行运行结果

9. “是否自动折行”属性

“是否自动折行”属性可控制是否自动换行,其值只能为“真”或“假”。默认情况下,该属性的值为“假”,即不会自动换行,如果标题文字长度超过标签宽度,只显示与标签等宽部分。

不折行与折行的效果,如图7-27所示。

如果要用标签组件显示大量文本,通常把“是否自动折行”属性置为“真”,并把“横向对齐方式”置为“0. 左对齐”。(当“是否自动折行”为“真”时,文本在纵向上自动顶对齐,“纵向对齐方式”属性不再生效。)

10. “数据源、数据列”属性

“数据源、数据列”属性通常在数据库操作中使用,用标签动态显示数据库中的数据。“数据源”属性用于指定一个与标签相连接的数据源组件的名称(该数据源组件通常应该拖放到程序界面中,它即代表了欲操作的数据库)。“数据列”属性用于指定数据源中的数据列,可以是列名或以1开始的列序号文本。

7.2.3 标签的应用例程

1. 电影字幕

使用标签模拟电影字幕的样子,文字从下方向上方缓缓移动。具体参见例程7-17,它加入了时钟组件,有兴趣的可以查看源代码,运行结果见图7-28,程序代码见图7-29。

具体参见例程7-18,该例程即用到了时钟组件,也用到纵向滚动条组件与图片框组件,结果是异曲同工。

2. 推拉字幕

具体参见例程7-19,运行该例程可看到标签文字由两个非常小的字逐渐放大,然后逐渐缩小,效果就像在推拉镜头,运行结果见图7-30,程序代码见图7-31。

在此,本例程为节约变量的申请,从而使用了标签1的“标记”属性作为一个变量使用,节约了一个变量的申请。但由于“标记”属性是文本型的,所以要转换为数值型,使用命令“到数值()”即可。

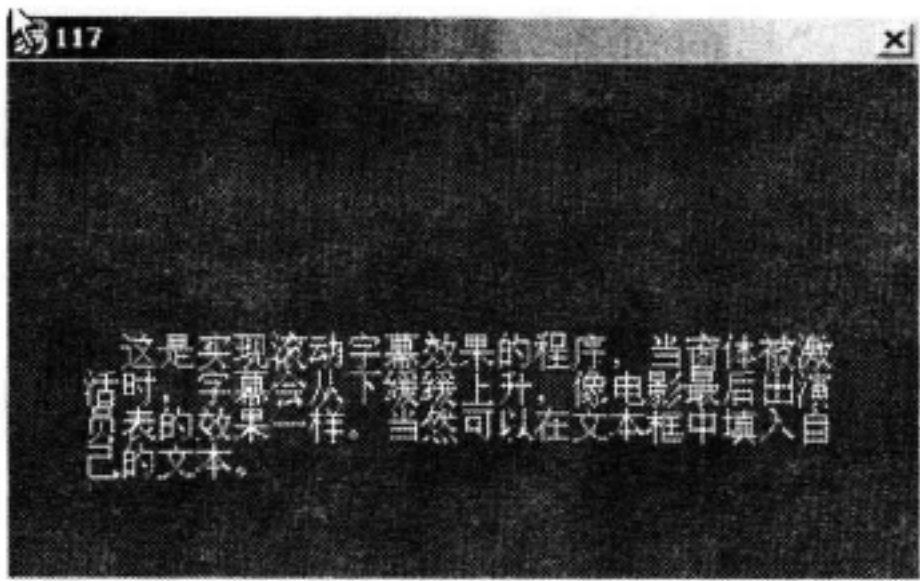


图 7-28 电影字幕效果运行结果

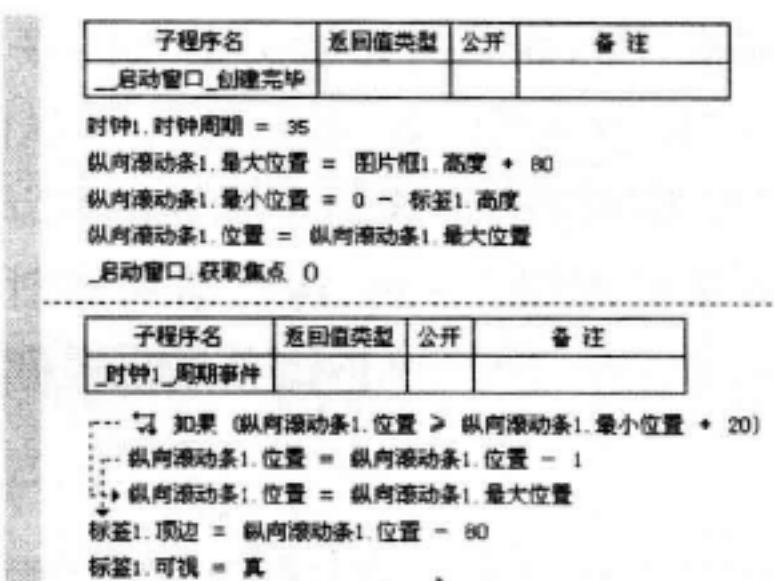


图 7-29 程序代码

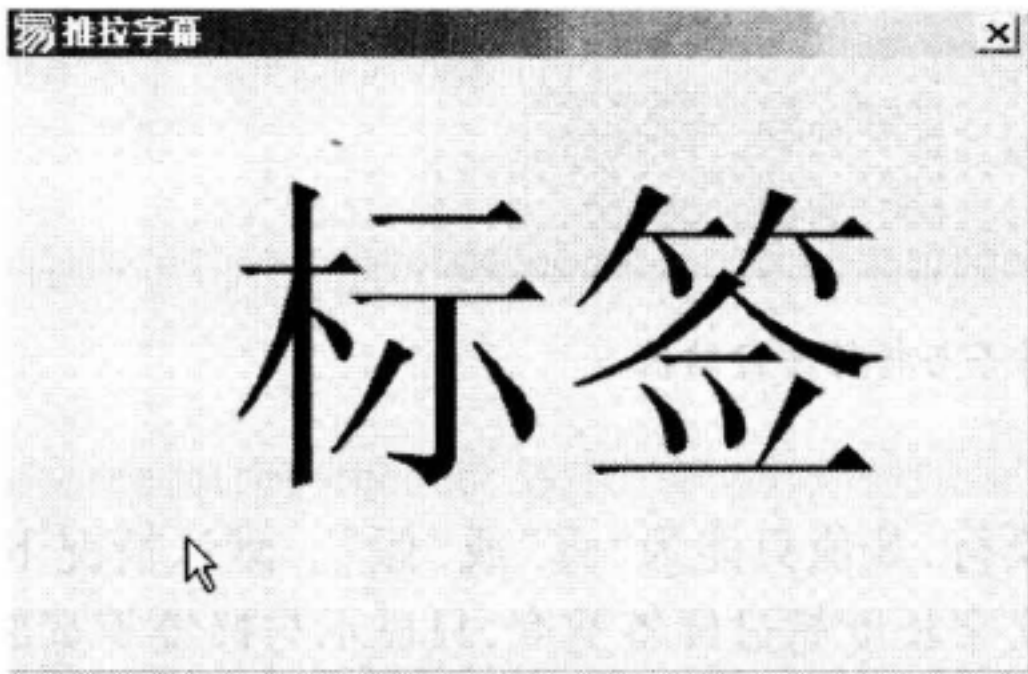


图 7-30 标签 - 推拉字幕运行结果

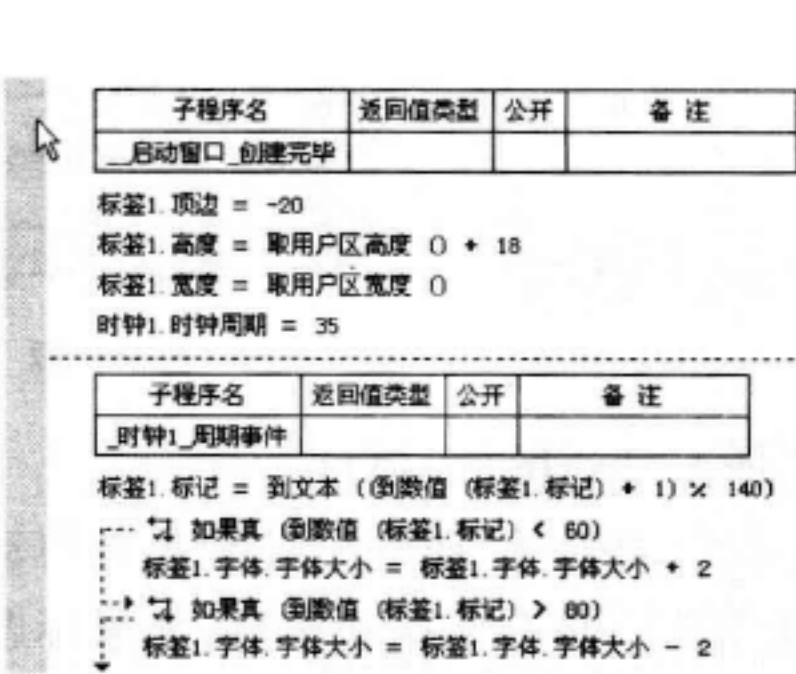


图 7-31 程序代码

3. 可移动的标签

具体参见例程 7-20,其运行结果如图 7-32 所示。

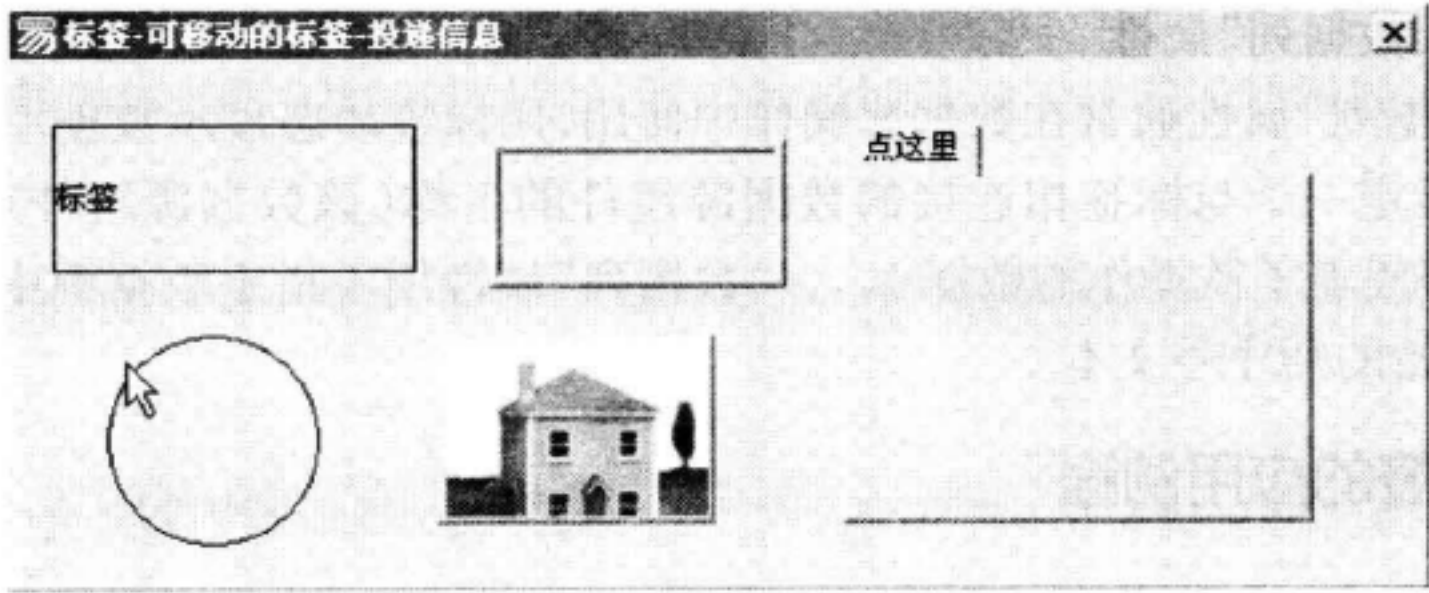






图 7-32 标签 - 可移动 - 投递信息运行结果

用鼠标移动这些组件,看到这些组件随着鼠标的拖动而移动了。除了标签组件可以被移动外,还有图片框组件、外形框组件、画板组件、选择夹组件等都可以移动,使用到的程序代码如图 7-33 所示。

4. 多彩进度条代码

具体参见例程 7-21,可以看到类似进度条的样子,运行结果如图 7-34 所示。

本例程使用了“刷新显示()”命令。这个命令与“获取焦点()”命令一样,前面加个对象就可以了,具体程序代码如图 7-35 所示。

窗口程序集名	保留	保留	备注
窗口程序集1			
子程序名	返回值类型	公开	备注
_标签1_鼠标左键被按下	逻辑型		
参数名	类型	参考	可空 数组 备注
横向位置	整型		
纵向位置	整型		
功能键状态	整型		
标签1. 投递信息 (161, 2, 0)			
子程序名	返回值类型	公开	备注
_图片框1_鼠标左键被按下	逻辑型		
参数名	类型	参考	可空 数组 备注
横向位置	整型		
纵向位置	整型		
功能键状态	整型		
图片框1. 投递信息 (161, 2, 0)			
子程序名	返回值类型	公开	备注
_选择夹1_鼠标左键被按下	逻辑型		
参数名	类型	参考	可空 数组 备注
横向位置	整型		
纵向位置	整型		
功能键状态	整型		
选择夹1. 投递信息 (161, 2, 0)			
子程序名	返回值类型	公开	备注
_外形框1_鼠标左键被按下	逻辑型		
参数名	类型	参考	可空 数组 备注
横向位置	整型		
纵向位置	整型		
功能键状态	整型		
外形框1. 投递信息 (161, 2, 0)			

图 7-33 程序代码

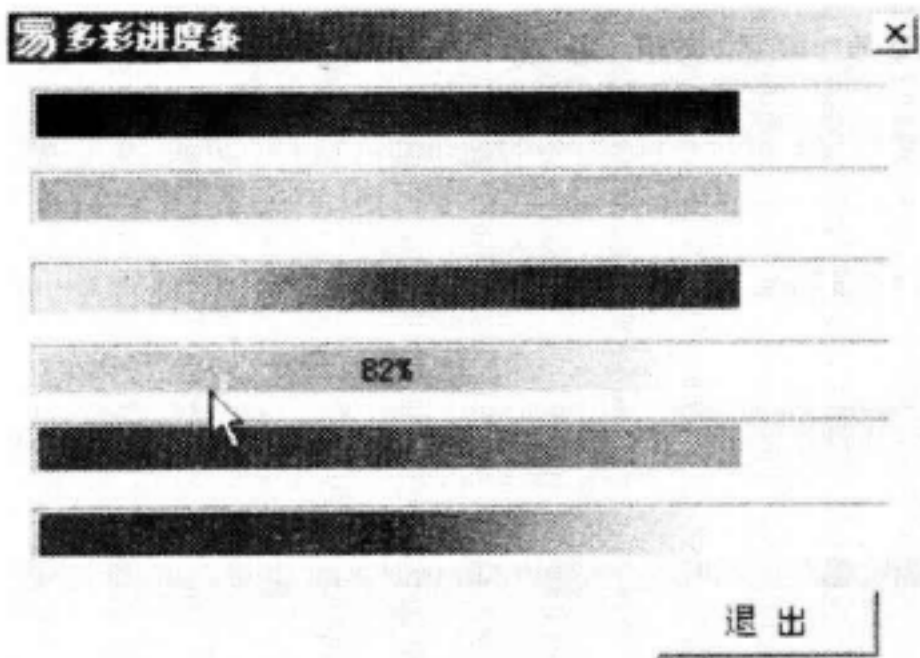


图 7-34 多彩进度条运行结果

子程序名	返回值类型	公开	备注
_时钟1_周期事件			
变量名	类型	静态	数组 备注
i		✓	
如果真 (i > 305)			
时钟1. 时钟周期 = 0			
标签1. 宽度 = i			
标签2. 宽度 = i			
标签3. 宽度 = i			
标签4. 宽度 = i			
标签5. 宽度 = i			
标签6. 宽度 = i			
标签4. 标题 = 到文本 (取整 (i ÷ 3.06)) * "%"			
标签6. 标题 = 到文本 (i)			
标签1. 刷新显示 0			
标签2. 刷新显示 0			
标签3. 刷新显示 0			
标签4. 刷新显示 0			
标签5. 刷新显示 0			
标签6. 刷新显示 0			
标签6. 获取焦点 0			
i = i + 1			

图 7-35 程序代码

5. 带有下拉条的标签

具体参见例程 7-22, 其运行结果如图 7-36 所示。

运行时可以看到, 当单击“资料夹”标签后, 下面的树型框就收起来了。再单击就会展开树型框。

在此使用到一个时钟组件, 还使用了一个标签 4 的组件, 标签 4 的作用即是高度拉长, 以覆盖树型框组件, 使人感觉上是树型框组件在收缩, 实际上是标签 4 覆盖的原因。大家可以试试, 直接将树型框的高度缩小会怎么样, 程序代码如图 7-37 所示。

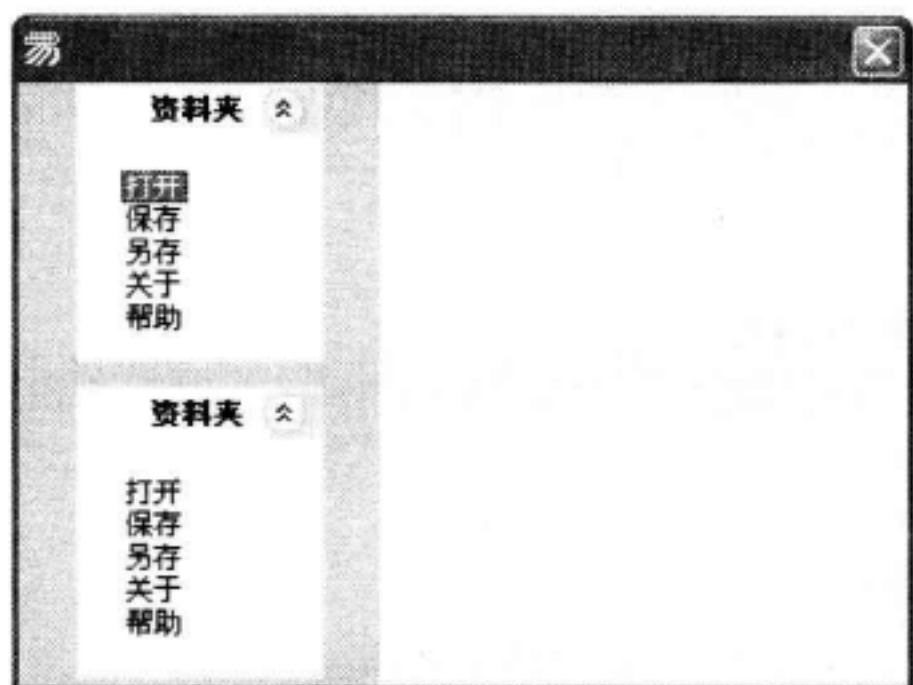


图 7-36 下拉条运行结果

子程序名	返回值类型	公开	备 注		
_标签1_鼠标左键被放开	逻辑型				
参数名	类 型	参 考	可 空	数 组	备 注
横向位置	整数型				
纵向位置	整数型				
功能键状态	整数型				

时钟1.时钟周期 = 10

子程序名	返回值类型	公开	备 注
_时钟1_周期事件			

```
如果 (展开 = 真)
    如果 (标签4.顶边 > 24)
        标签4.顶边 = 标签4.顶边 - 4
        标签4.高度 = 标签4.高度 + 4
        标签1.底图 = #下箭头图
        时钟1.时钟周期 = 0
        展开 = 假
    如果 (标签4.顶边 < 120)
        标签4.顶边 = 标签4.顶边 + 4
        标签4.高度 = 标签4.高度 - 4
        标签1.底图 = #上箭头图
        时钟1.时钟周期 = 0
        展开 = 真
```

图 7-37 程序代码

例程 7-22 是直接将树型框隐藏起来。这个例子非常简单,只是将树型框组件的显示属性改了一下,程序代码如图 7-38 所示。

子程序名	返回值类型	公开	备 注		
_标签3_鼠标左键被放开	逻辑型				
参数名	类 型	参 考	可 空	数 组	备 注
横向位置	整数型				
纵向位置	整数型				
功能键状态	整数型				

```
树型框2.可视 = 取反 (树型框2.可视)
如果 (标签3.底图 = #上箭头图)
    标签3.底图 = #下箭头图
    标签3.底图 = #上箭头图
```

图 7-38 程序代码

6. 动态设置标签的大小

有的读者设定了标签的字体和宽度,使其正好显示“易语言”3个字,但在其他计算机中运行时却发现标签的宽度变小了,不能显示所有内容,这个问题产生的原因在于不同的系统中,相同字体其默认的宽度和高度会有一些误差。

具体解决方法如下:在窗口中添加一个标签组件,设定好字体及标题(如“易语言”),然后再添加一个画板组件,设置其“可视”属性为“假”,其字体与标签的字体一样。然后在窗口创建完毕事件处理子程序中输入如下代码:

```
标签 1. 宽度 = 画板 1. 取宽度 (标签 1. 标题)
```


7.3 画板组件

7.3.1 画板概述

画板组件是一种包容型窗口单元,用来在上面显示和绘制图片、形状、文字等。也用来分割窗口的区域,放置不同类型的组件,起到分类的作用。另外还可以作为一般组件的背景或是信息栏、状态栏。

画板组件在易语言中十分重要,功能十分强大,充分发挥你的想象力,可以实现意想不到效果。初学者只要掌握最基本的用法就行了。

画板组件的主要属性有“自动重画、画笔颜色、刷子颜色、画笔类型、刷子类型、画出方式、画笔粗细、绘画单位、文本颜色、文本背景颜色、字体”等。

画板组件的重要方法有“画图片()、取图片()、取图片宽度()、取图片高度()、写文本行()、滚动写行()、写出()、定位写出()、画点()、画直线()、画矩形()、画圆角矩形()、画渐变矩形()、画多边形()、画椭圆()、画弧线()、画弦()、画饼()、填充矩形()、反转矩形区()、取宽度()、取高度()、取点()、取设备句柄()、复制()、置写出位置()、清除()”等。

画板组件的主要事件有“绘画”。

7.3.2 画板的属性

1. “底图”、“底图方式”、“边框”、“画板背景色”属性

“底图”属性中可以导入一张图片。“底图方式”属性中可以设置图片的显示状态,如“0. 图片居左上”、“1. 图片平铺”、“2. 图片居中”。

“边框”属性可以为画板加一个边框。

“画板背景色”属性可为画板事先加入一种颜色。

具体参见例程7-23,其运行结果如图7-39所示。



图7-39 底图 & 底图方式运行结果

如果预先加载了底图在画板中,执行“画板1. 清除()”命令没有用,但画板中却仍显示原来的图片。这是因为画板的“清除”命令只能清除在画板上写或画出的内容,不能清除画板初始设置好的底图。要清除画板的底图,代码如下:

```
画板1. 底图 = { }
```

2. “自动重画”属性

“自动重画”属性可控制画板是否自动重画其中的内容,逻辑型,只能为“真”或“假”,默

认为“假”，即不自动重画。但在使用中，通常把该属性设为“真”。

下面通过一个试验来进一步了解“自动重画”的含义：在设计窗口放一个画板组件，保持默认属性；再放一个按钮，在按钮的“被单击”事件中向画板画图；运行程序，单击按钮后，画板中出现画图；可一旦画板被其他窗口遮住，里面的内容就消失了，这是因为画板的“自动重画”属性默认为“假”的缘故，如果设置“自动重画”属性为“真”，就不会出现这种现象了。

如果保持“自动重画”属性为“假”（默认值），一般都需要响应画板的“绘画”事件；通常为了方便，把它设为“真”。

画板的其他属性不再一一讲解。

具体参见例程 7-24，运行结果如图 7-40 所示。

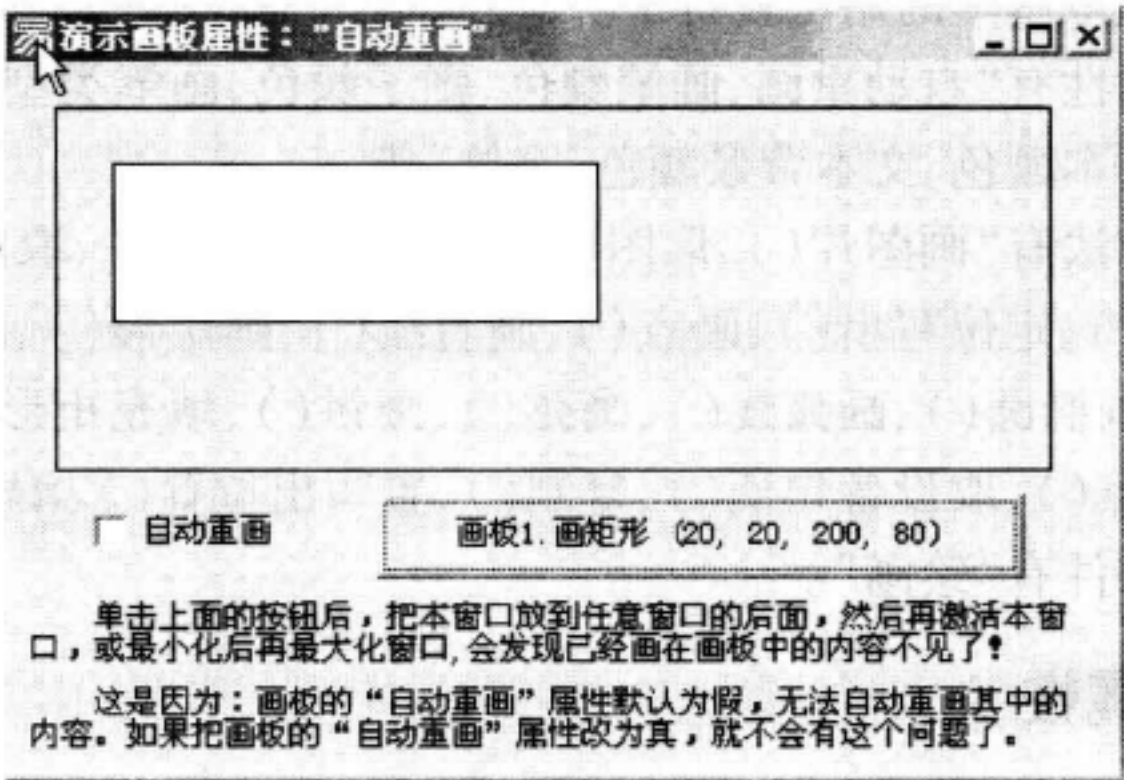


图 7-40 画板属性 - 自动重画运行结果

7.3.3 画板的方法

画板的主要的方法有“画图片()、取图片()、取图片宽度()、取图片高度()、写文本行()、滚动写行()、写出()、定位写出()、画点()、画直线()、画矩形()、画圆角矩形()、画渐变矩形()、画多边形()、画椭圆()、画弧线()、画弦()、画饼()、填充矩形()、反转矩形区()、取宽度()、取高度()、取点()、取设备句柄()、复制()、置写出位置()、清除()”等。

画板的方法很多，参数也通常较多，此处无法一一列出。另外，除了“画图片()”、“清除()”等少数方法外，其他都不大常用。

画板组件的方法与参数很多，难于记忆，在此教大家一个技巧：要使用哪个“方法”，先输入它的名称后回车（比如要画一条直线，先打上“画板 1. 画直线()”回车），然后将光标移到这条命令上，按键盘上的 Alt 键和右方向键，这时易语言会在该行下面分行列出所有的参数名称，可向每个参数填充值，这样输入参数就方便多了，如图 7-41 所示。

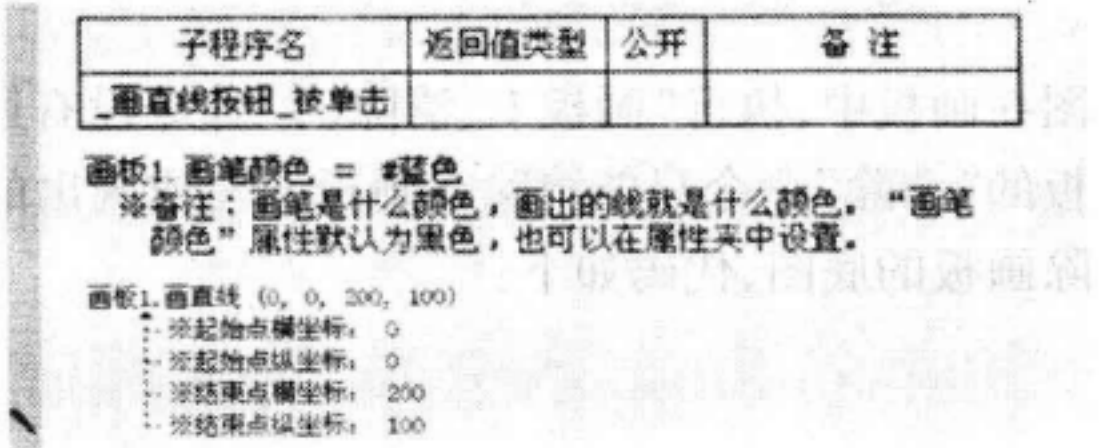


图 7-41 展开命令

1. “画图片()”方法

具体参见例程 7-25,运行结果如图 7-42 所示。

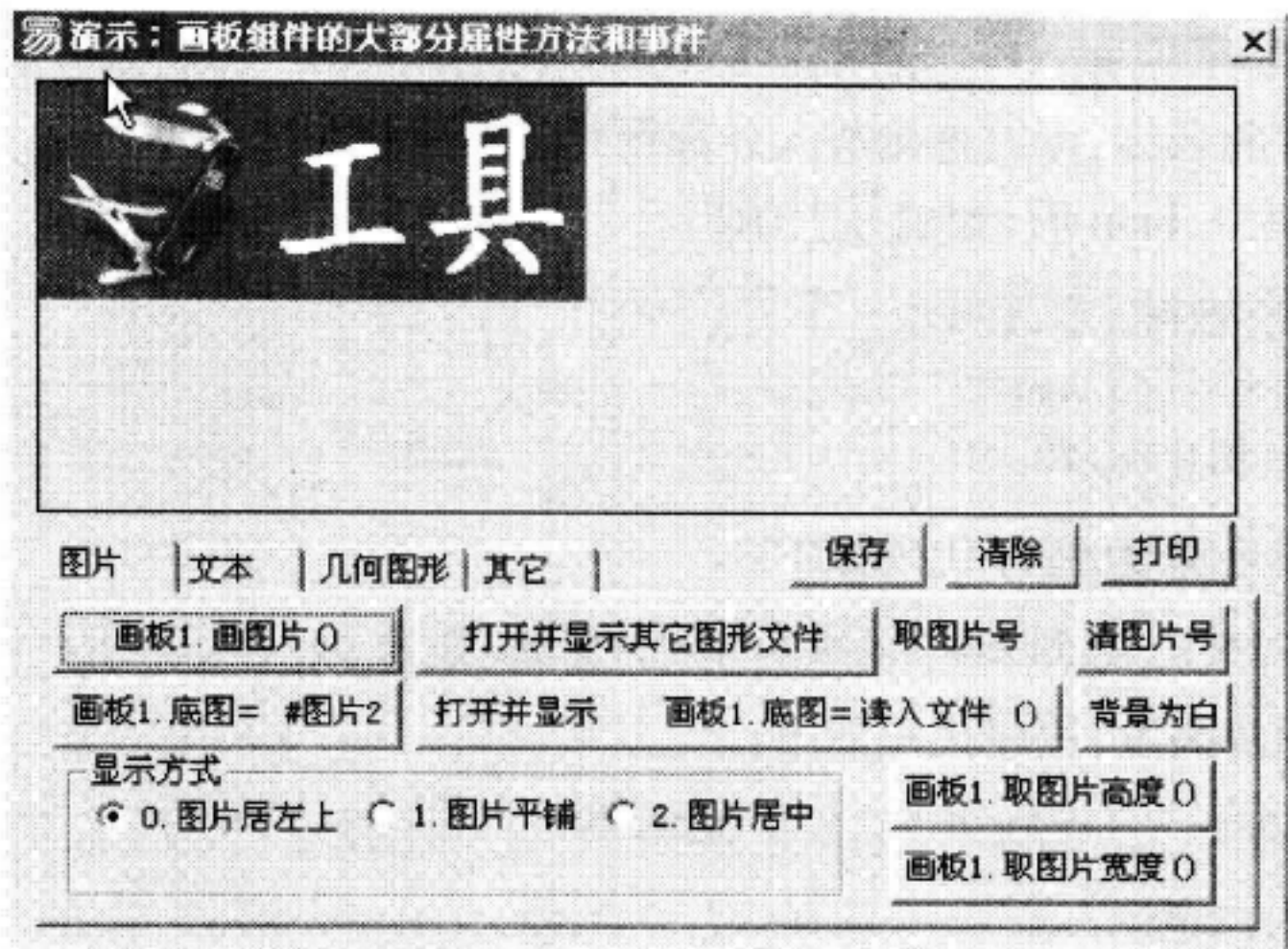


图 7-42 画图方法运行结果

使用这个方法之前,一定要事先设置好一个程序集变量“图片号”,此图片号非常重要,将在本窗口中一直使用到。程序中有两个图片资源,还有一个通用对话框组件。

下面的程序代码是第 1 排第 1 个按钮,用于载入图片资源中的图片。

```
图片号 = 载入图片 (#图片 1)
画板 1. 画图片 (图片号, 0, 0, , , )
```

第 2 排第 1 个按钮也是用于载入图片资源中的图片。

```
画板 1. 底图 = #图片 2
```

载入图片资源中的图片也可以这样写代码,但是这样作就取不到图片号,如果有打印的功能,将无法将图片号传送给打印机组件来处理。因此最好使用第 1 种方法。若程序中不要求打印,才可以用第 2 种方法。两种方法的区别如表 7-1 所示。

表 7-1 两种方法简单对比

方 法	显 示 方 式	取图片高度及宽度	打 印 机
用图片号的方法,间接引用	不响应	可以	可用
不用图片号的方法,直接引用	响应	取不到	不可用

【注意】在第 2 种方法中,如果同时使用第 1 种方法,产生了图片号也可以实现打印。

第 1 排第 2 个按钮是打开电脑中的其他图形文件显示到画板中,程序代码如下:

```
如果真 (通用对话框 3. 打开 ())
    图片号 = 载入图片 (通用对话框 3. 文件名)
    画板 1. 画图片 (图片号, 10, 10, , , )
如果真结束
```

第 2 排第 2 个按钮演示了使用图片号的方法,如果直接引用打开外部图片,程序代码如下:

如果真 (通用对话框 3. 打开 ())

画板 1. 底图 = 读入文件 (通用对话框 3. 文件名)

如果真结束

根据外部图片的名称,可以直接载入外部图片,程序代码如下:

示例:载入图片 1. bmp 后,在画板上画出。

局部变量:图片号 数据类型:整数型

图片号 = 载入图片 (“1. bmp”)

画板 1. 画图片 (图片号, 10, 10, , ,)

取画板 1 的图片宽度,程序代码如下:

信息框 (“图片宽度:” + 到文本 (画板 1. 取图片宽度 (图片号)), 0,)

取画板 1 的图片高度,程序代码如下:

信息框 (“图片高度:” + 到文本 (画板 1. 取图片高度 (图片号)), 0,)

【注意】背景为白色。

画板 1. 画板背景色 = #白色

显示方式的 3 个单选框的被单击事件的程序代码分别如下:

画板 1. 底图方式 = 0

画板 1. 底图方式 = 1

画板 1. 底图方式 = 2

还可以通过单击名为:“取图片号”的标签,而取到图片号,如果图片号为“0”,那么表示没有图片号,没有图片号就不能知道图片的高度与宽度,也不能打印。“_标签 8_鼠标左键被按下”的程序代码如下:

标签 8. 标题 = 到文本 (图片号)

2. 写文本类方法

画板 1. 滚动写行 (编辑框 1. 内容)

画板 1. 滚动写行 (12345, 取现行时间 ())

在当前写出位置写出指定的文本、数值、逻辑值或日期时间,并将现行写出位置调整到下行行首。如果现行画板高度无法容纳当前所要写出的行,则自动向上滚动画板内容。最后一个参数可以被重复添加。

画板 1. 写文本行 (编辑框 1. 内容)

画板 1. 写文本行 (12345, 取现行时间 ())

在当前写出位置写出指定的文本、数值、逻辑值或日期时间,并将现行写出位置调整到下行行首。如果现行画板高度无法容纳当前所要写出的行,则自动向上滚动画板内容。最后一个参数可以被重复添加。

方法“写文本行”与“滚动写行”的用法是相同的,区别在于:如果画板只能容纳 3 行文本,用“写文本行”写出的第 4 行及以后的文本,都到画板外面去了;而“滚动写行”呢,顾名思义,如果现行画板高度无法容纳当前所要写出的行,则自动向上滚动画板内容,刚刚写出的文本行永远能够看到(至少能够看到一部分——如果刚写出的内容超过 3 行的话)。

由于不必过多地考虑显示问题。所以经常使用“滚动写行”。

“定位写出”可以写到指定的位置上。

画板 1. 定位写出 (文本到数值 (X 编辑框. 内容), 文本到数值 (Y 编辑框. 内容), “‘定位写出’内容”, 123, 取现行时间 ())

在点(x,y)处写文本。不改变现行写出位置。简单用法如下:

画板 1. 定位写出(3,5,“文本”) //在点(3,5)处写文本。

画板 1. 置写出位置 (文本到数值 (X 编辑框. 内容), 文本到数值 (Y 编辑框. 内容))
紧跟上次的操作后面写出的程序代码如下:

画板 1. 写出 (“‘写出’内容”, 123, 取现行时间 ())

设置好字体后再写出的代码如下:

如果真 (通用对话框 2. 打开 ())

字体标签. 字体. 加粗 = 通用对话框 2. 加粗

字体标签. 字体. 倾斜 = 通用对话框 2. 倾斜

字体标签. 字体. 删除线 = 通用对话框 2. 删除线

字体标签. 字体. 下划线 = 通用对话框 2. 下划线

字体标签. 字体. 字体名称 = 通用对话框 2. 字体名称

字体标签. 字体. 字体大小 = 通用对话框 2. 字体大小

字体标签. 文本颜色 = 通用对话框 2. 字体颜色

画板 1. 字体 = 字体标签. 字体

画板 1. 文本颜色 = 通用对话框 2. 字体颜色

如果真结束

如果想要改变要写字的前景色与背景色,可以使用以下程序代码:

字体标签. 背景颜色 = 颜色选择器 1. 颜色

画板 1. 文本背景颜色 = 颜色选择器 1. 颜色

3. 画几何图形类方法

由于画点与画线的内容形式差不多,在此只看一种,其他功能可以通过双击按钮查看源代码,运行结果如图 7-43 所示。

画板 1. 画点 (8, 9, #白色)

画板 1. 画直线 (0, 0, 200, 100)

画板 1. 画矩形 (20, 30, 200, 100)

画板 1. 画弧线 (10, 10, 300, 150, 0, 0, 180, 160)

画板 1. 画圆角矩形 (10, 10, 395, 150, 30, 30)

画板 1. 画椭圆 (20, 5, 160, 145)

画板 1. 画弦 (10, 10, 300, 150, 0, 0, 180, 160)

画板 1. 画饼 (10, 10, 300, 150, 0, 0, 180, 160)

画板 1. 填充矩形 (0, 0, 画板 1. 画板宽度, 画板 1. 画板高度)

画板 1. 翻转矩形区 (0, 0, 画板 1. 画板宽度, 画板 1. 画板高度)

标签 6. 背景颜色 = 画板 1. 取点 (100, 100)

画板 1. 复制 (0, 0, 画板 2. 画板宽度, 画板 2. 画板高度, 画板 2, 0, 0,)

以上的方法都可以在支持库面板中“系统核心支持库”的“数据类型”中找到,如图 7-44 所示。

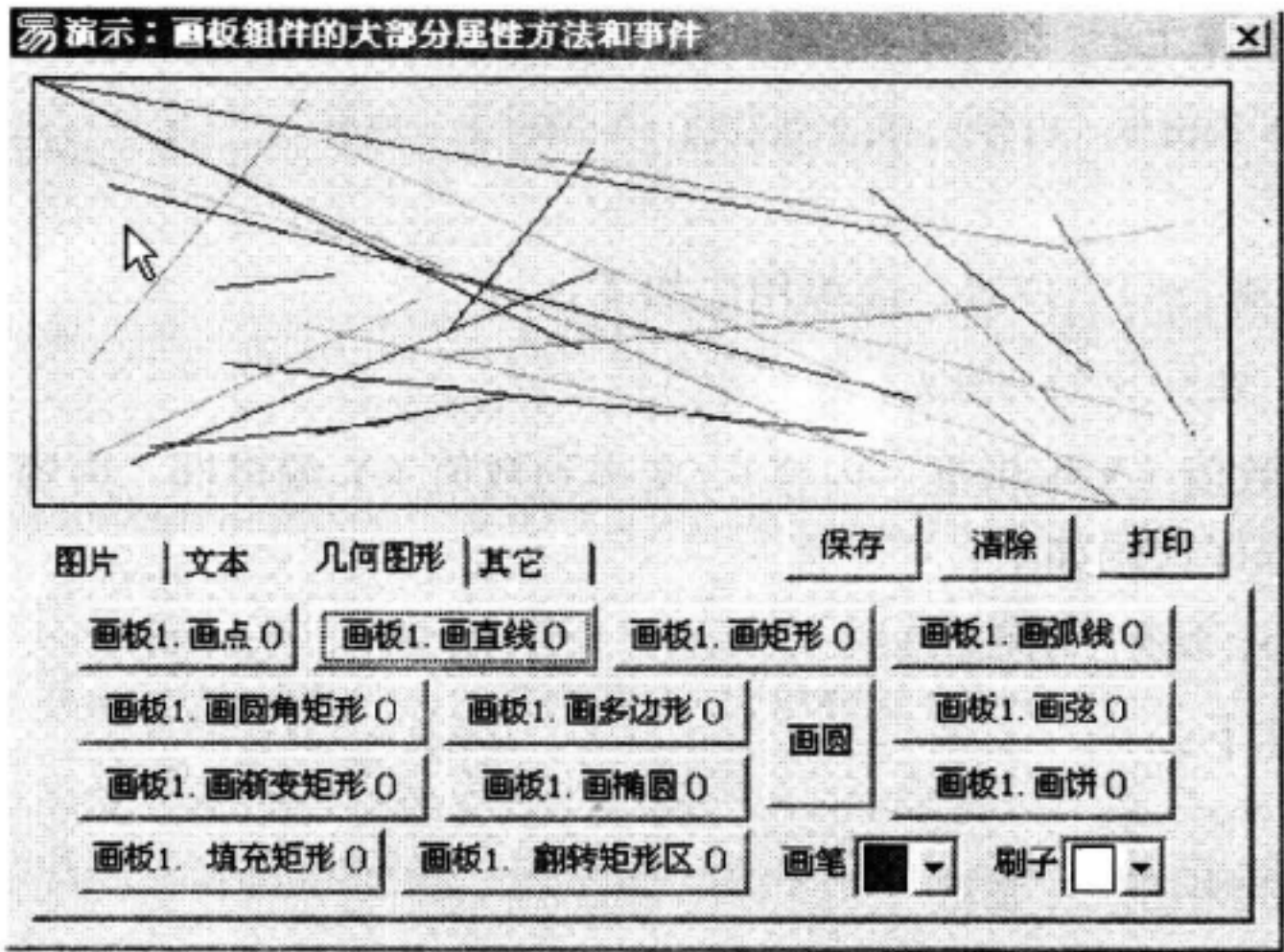


图 7-43 画板组件的大部分属性方法和事件运行结果



图 7-44 支持库面板

4. 保存图片类方法

怎样才能把画板中的内容保存为图片文件？用以下代码可实现：

写到文件(文件名,画板 1. 取图片())

如果保存时要改变图片的大小,则可以用以下代码：

写到文件(文件名,快照(画板 1. 取窗口句柄(),图片宽,图片高))

5. 打印图片类方法

打印按钮使用的 3 行程序代码如下：

打印机 1. 开始打印(真,真,,)

打印机 1. 画图片(载入图片(画板 1. 取图片(,)),1,1,画板 1. 取图片宽度(图片号) × 3,画板 1. 取图片高度(图片号) × 3,)

打印机 1. 结束打印()

将画板的单位设置为逻辑单位,如 0.1mm,然后调用“取图片宽度”和“取图片高度”取回图片的逻辑尺寸值,再在打印机的“画图片”方法中作为尺寸参数提供(打印机的绘画单位必须一致),系统会自动缩放。

7.3.4 画板的事件

画板的重要事件只有一个：“绘画”。

“绘画”事件的产生时机:当画板中的全部或一部分区域需要被重新绘制时产生此事件。

该事件的处理子程序有 4 个参数,分别提供目前画板中需要更新的矩形区域的左上角、右下角坐标。使用中通常可忽略这些参数(即一般不使用它们)。

如果设置了画板的“自动重画”属性为“假”(默认值),则所有对画板的绘图操作,通常在本事件中进行。这样当画板被其他窗口遮住后又显示时,程序自动产生“绘画”事件,自动调用“绘画”事件的“事件处理子程序”,重新绘制画板中的内容,以确保画板中的内容永远不会消失。

具体参见例程 7-26,其运行结果如图 7-45 所示。

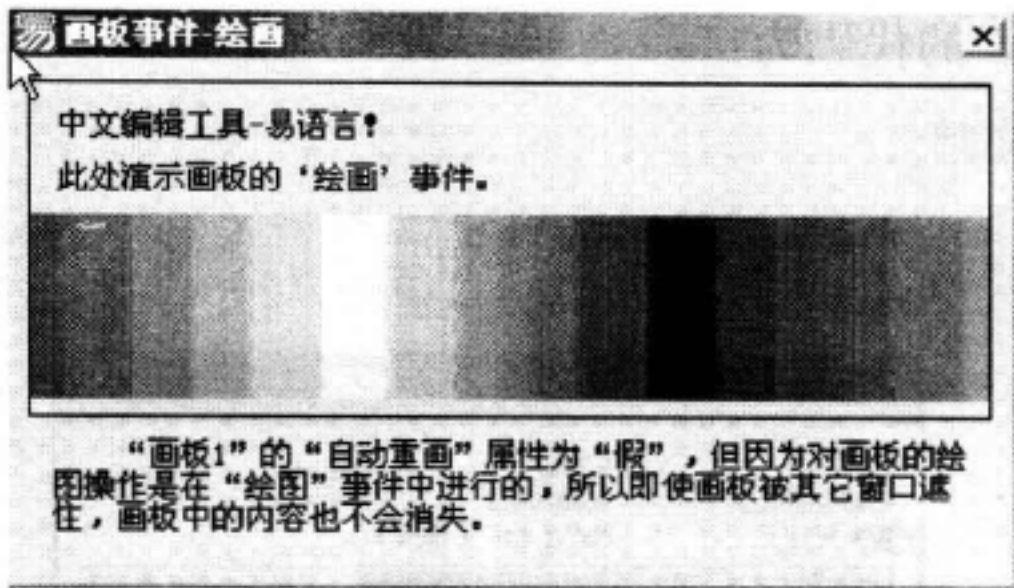



图 7-45 画板事件 - 绘画运行结果

画板 1 的“自动重画”属性为“假”,但因为对画板的绘图操作是在“绘图”事件中进行的,所以即使画板被其他窗口遮住,画板中的内容也不会消失。

画板用好了可以使界面非常漂亮,但总要一笔一笔的画,可视化几乎没有,所以要做得漂亮一些,往往需要试验很多次。


7.4 图片框组件

7.4.1 图片框组件概述

图片框的主要作用就是显示图片、用于装饰程序界面。图片框重要属性有:“图片”、“边框”、“背景颜色”、“显示方式”、“播放动画”、“数据源”、“数据列”。图片框组件没有重要方法。

7.4.2 图片框组件的重要属性

1. “图片”属性可指定图片框组件要显示的图片,该功能支持 BMP、JPG、GIF、ICO 图片格式,有三种提供图片的方式:

(1)单击属性“图片”右侧按钮,在弹出的图片属性对话框中选择“更换”按钮,找到相应的图片文件,直接加载图片到此属性中,如图 7-46 所示。

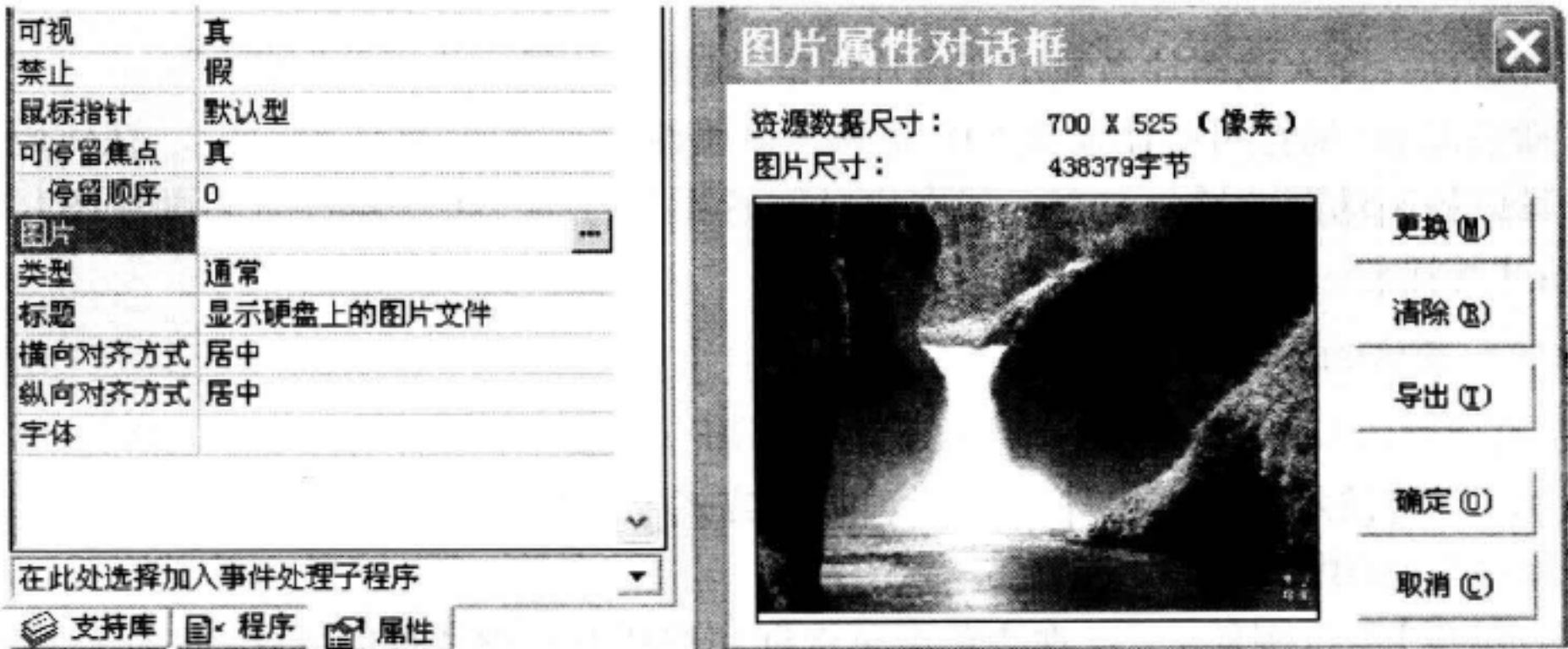


图 7-46 图片属性对话框

(2) 将一张图片存到易语言的“图片资源”中,如图 7-47 所示,再作为资源直接引用,引用图片资源名称为“图片 1”的代码为:

“图片框 1. 图片 = #图片 1”



图 7-47 图片资源属性

(3) 在程序运行期间读取硬盘上的一个图片文件,例如:

图片框 1. 图片 = 读入文件(通用对话框 1 文件名)

2. “显示方式”属性,用来控制图片框中图片的显示方式。类型为整数型,有 3 个可选值“0. 图片居左上”、“1. 缩放图片”、“2. 图片居中”,默认为“0. 图片居左上”。在这里,它有一个特殊的显示方式,即“缩放图片”显示方式。当图片框组件的高度与宽度变化时,图片框中的图片会适应图片框的变化进行相应的缩小或放大,填充整个图片框。当选中“缩放图片”单选框时,可以看到,图片填充了整个图片框。无论图片框的高度与宽度如何变化,都会自动缩放填充到图片框中的。例如:

“图片框 1. 图片 = 读入文件(通用对话框 1 文件名)”。

图片框 1. 显示方式 = 1

程序执行后,读入该图片名显示在图片框中,设置图片显示方式为“缩放”。

3. “播放动画”属性可控制播放 GIF 动画。该属性为逻辑型,默认为“真”。在代码中设置该属性,可以控制动画的播放与否。如在设计时置本属性为“真”,运行时动画在图片框被创建后自动开始播放。例如:

图片框 1. 播放动画 = 真

程序执行后,自动开始播放“图片框 1”中的 GIF 动画。

【范例 7-1】演示图片框组件“图片”、“显示方式”、“播放动画”三个属性。具体参考例程 7-27。程序代码如图 7-48 所示。

【运行结果】运行例程 7-1,观测单击单选框和选择框时图片框的变化,如图 7-49 所示。

【注意】播放动画属性只在“图片”属性为 GIF 格式时有效。当单击“播放 GIF 动画”选择框时,如果勾选,即为播放动画,如果没有勾选,则不播放动画。

窗口程序集名	保留	保留	备注
窗口程序集1			

子程序名	返回值类型	公开	备注
_按钮1_被单击			

图片框1.图片 = #图片1

子程序名	返回值类型	公开	备注
_单选框1_被单击			

图片框1.显示方式 = 0 ' 图片居左上

子程序名	返回值类型	公开	备注
_单选框2_被单击			

图片框1.显示方式 = 1 ' 缩放图片

子程序名	返回值类型	公开	备注
_单选框3_被单击			

图片框1.显示方式 = 2 ' 图片居中

子程序名	返回值类型	公开	备注
_选择框1_被单击			

如果 (选择框1.选中 = 真)

 图片框1.播放动画 = 真

 图片框1.播放动画 = 假

子程序名	返回值类型	公开	备注
_启动窗口_创建完毕			

XP风格 (#蓝色风格)

图 7-48 图片框属性演示代码

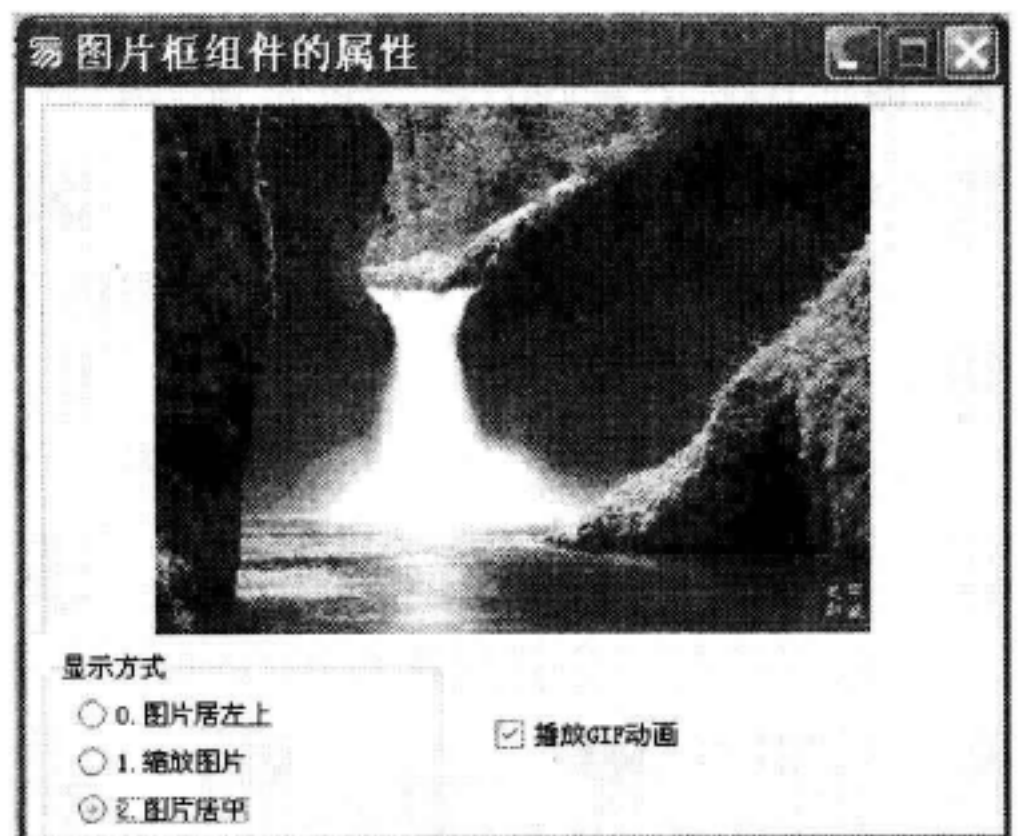



图 7-49 图片框属性演示运行结果

7.5 外形框组件

7.5.1 外形框组件概述

外形框主要用于显示矩形、正方形、椭圆、圆、圆角矩形、圆角正方形、横向线和纵向线这8种形状,其主要属性有“外形”、“线型”、“线宽”、“线条效果”、“线条颜色”、“填充颜色”、“背景颜色”等。

7.5.2 外形框组件属性

1. “外形”属性功能:可以在窗体、图片框或分组框等容器内使用形状控件,但是不能用形状控件作为容器,因为它不是包容型窗口单元。“外形”属性用来设置其形状,可以在属性窗口设置,如图7-50 直接在属性窗口设置外行为椭圆形。

也可以在运行程序时用代码加载,例如:

```
外形框1.外形 = 2
```

程序执行后会将外形框的外形设置为椭圆形。

2. “背景颜色”属性

本属性用于设置外形框内的背景颜色。属性值为 RGB 颜色值,可以通过“取颜色值”调

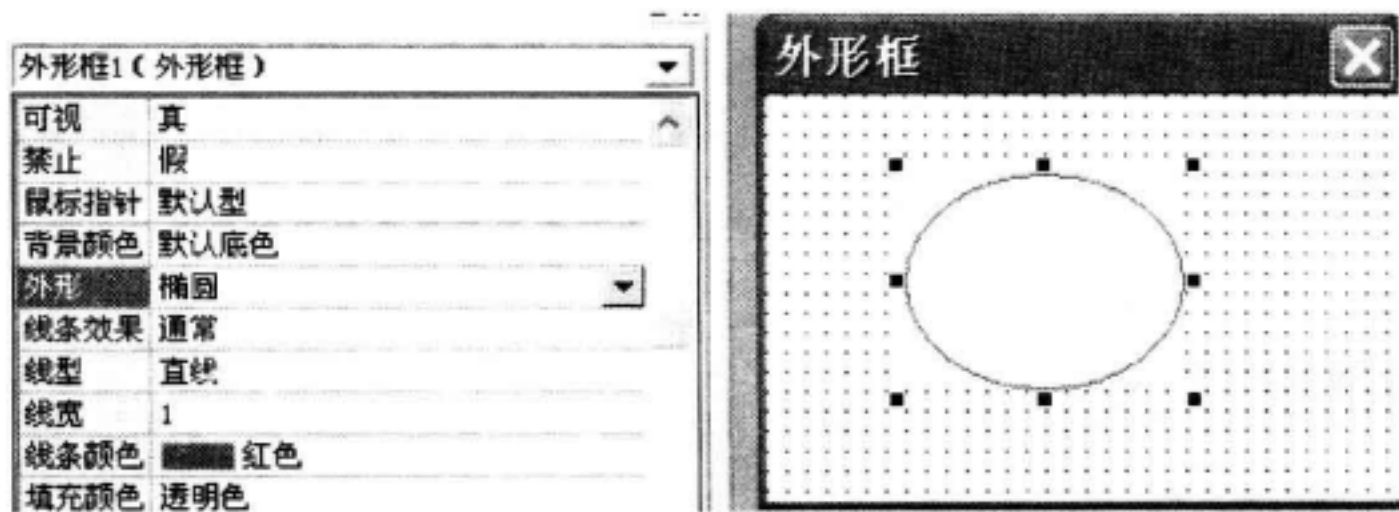


图 7-50 外形框外形属性

配,或使用易语言内置的颜色常量,如“#红色”等。例如:

外形框_背景.背景颜色 = #红色

程序执行后会将外形框的背景颜色设置为红色。

3. “线条效果”属性

本属性用于指定外形框线条效果。例如:

外形框_效果.线条效果 = 2

程序执行后会将外形框的线条效果设为凸出样式。

4. “线型”属性

本属性用于设置外形框的线条类型,例如:

外形框_线型.线型 = 2

程序执行后会将外形框的边框线型设为划线样式。

5. “线宽”属性

本属性用于设置外形框线条的宽度,单位为像素。但仅在线条效果为“通常”且线型为“直线”时该属性才有效。例如:

外形框_线宽.线宽 = 10

程序执行后会将“外形框的线条宽度设为 10 像素。

6. “线条颜色”属性

本属性用于设置外型框线条的颜色。属性值为 RGB 颜色值,可以通过“取颜色值”调配,或使用易语言内置的颜色常量,如“#红色”等。但仅在线条效果为“通常”时本属性才有效。例如:

外形框_线条.线条颜色 = #绿色

程序执行后会将外形框的线条颜色设为绿色。

7. “填充颜色”属性

本属性用于设置外形框中图形范围的填充颜色。属性值为 RGB 颜色值,可以通过“取颜色值”调配,或使用易语言内置的颜色常量,如“#红色”等。但仅在外形为线条时本属性才有效。例如:

外形框_填充色 = #绿色

程序执行后将外形框_填充为绿色

【范例7-2】外形框属性的演示。要求设计外型框的背景颜色为红色,通过单击鼠标可以变换外形框的不同外形,线宽为10,线条颜色是蓝色,内部填充色是绿色。具体参考例程7-28。

详细的程序代码如图7-51所示。

子程序名	返回值类型	公开	备注
按钮_背景颜色_被单击			

外形框_变换内部形态.背景颜色 = #红色 可以设置外形框背景的颜色,这里用颜色的常量值来显示背景颜色

子程序名	返回值类型	公开	备注
按钮_外形_被单击			

外形框-外形的属性有7种,从0-7-通过判断,可以每次点击一次外形按钮,就可变换一种外形

如果真 (外形框_变换内部形态.外形 = 0)

外形框_变换内部形态.外形 = 1

返回 0

如果真 (外形框_变换内部形态.外形 = 1)

外形框_变换内部形态.外形 = 2

返回 0

如果真 (外形框_变换内部形态.外形 = 2)

外形框_变换内部形态.外形 = 3

返回 0

如果真 (外形框_变换内部形态.外形 = 3)

外形框_变换内部形态.外形 = 4

返回 0

如果真 (外形框_变换内部形态.外形 = 4)

外形框_变换内部形态.外形 = 5

返回 0

如果真 (外形框_变换内部形态.外形 = 5)

外形框_变换内部形态.外形 = 6

返回 0

如果真 (外形框_变换内部形态.外形 = 6)

外形框_变换内部形态.外形 = 7

返回 0

如果真 (外形框_变换内部形态.外形 = 7)

外形框_变换内部形态.外形 = 0

返回 0

子程序名	返回值类型	公开	备注
按钮_线宽_被单击			

用一个编辑框作为接收线宽的数值,只要在编辑框内输入数值,就可以对外形框的线宽做出变化。

外形框_变换内部形态.线宽 = 到数值 (编辑框_线宽值.内容)

子程序名	返回值类型	公开	备注
按钮_线条颜色_被单击			

外形框_变换内部形态.线条颜色 = #蓝色

子程序名	返回值类型	公开	备注
按钮_填充颜色_被单击			

外形框_变换内部形态.填充颜色 = #绿色

图7-51 外形框属性演示代码

【运行结果】运行例程7-2,观测单击各个按钮时图片框的变化,如图7-52所示。

【注意】有多个属性条件的可以通过判断进行,每点击一次按钮就会显示一种方法及效果。

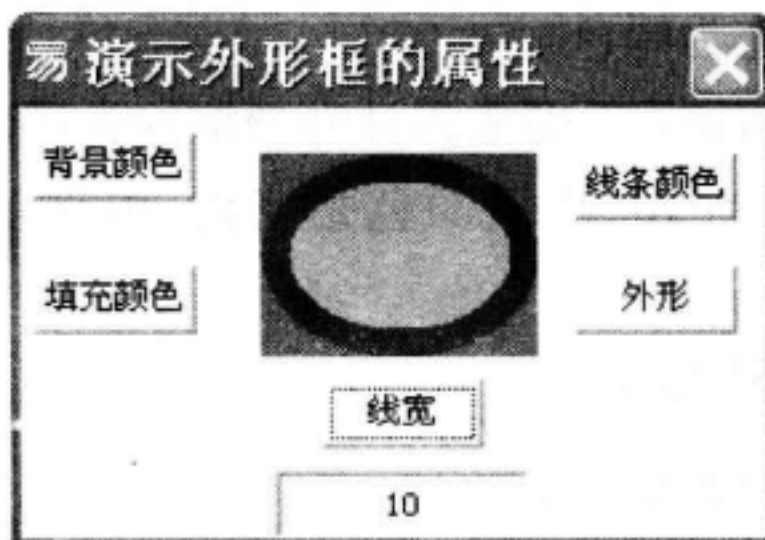



图 7-52 外形框属性运行结果

7.6 影像框组件

7.6.1 影像框组件概述

影像框组件可用来播放 AVI 格式的动画光标文件。影像框的用法和图片框差不多,不同之处有两点:一是影像框显示图片的时候,耗费的系统资源比图片框要少,而且它的绘图速度也比图片框快;二是图片框是一种容器型控件,它里面还可以放其他控件,移动图片框,里面的控件也会随之移动,而影像框就不一样了。

影像框的重要属性有“文件名”、“居中播放”、“透明背景”、“播放”、“播放次数”属性等。影像框没有重要方法,也没有专有事件。

7.6.2 影像框组件属性

1. “文件名”属性

本属性用于指定欲播放影像文件的名称。例如:

影像框. 文件名 = “D:\clock.avi”

2. “居中播放”属性

本属性用于指定在播放影像时影像是否居中。“真”为居中,“假”为居左上。

影像框. 居中播放 = 假

3. “背景透明”属性

本属性用于指定播放影像时影像框背景是否透明。“真”为透明,“假”为不透明。本属性在播放多数影像时效果不是很明显,但仍建议设置为“真”。例如:

影像框. 背景透明 = 真

4. “播放”属性

如果指定了有效的影像文件,通过改变此属性可控制其播放或停止。如在设计时置本属性为“真”,运行时影像文件在影像框被创建后即自动开始播放。例如:

影像框. 播放 = 真

5. “播放次数”属性

本属性用于指定影像的播放次数。若为-1,则无限次地循环播放。例如:

影像框. 播放次数 = -1

【范例7-3】影像框组件属性的演示。具体参考例程7-29。

详细的程序代码如图7-53所示。

【运行结果】运行例程7-3,观测单击各个按钮时影像框的变化,如图7-54所示。

窗口程序集名	保留	保留	备注
窗口程序集1			

子程序名	返回值类型	公开	备注
_按钮_浏览_被单击			

通用对话框1.初始目录 = 取运行目录 ()
 如果其 (通用对话框1.打开 ())
 编辑框_地址.内容 = 通用对话框1.文件名
 影像框_显示.文件名 = 通用对话框1.文件名

子程序名	返回值类型	公开	备注
_按钮_播放_被单击			

影像框_显示.播放 = 真

子程序名	返回值类型	公开	备注
_按钮_停止_被单击			

影像框_显示.播放 = 假

图7-53 影像框属性演示代码



图7-54 影像框属性演示运行结果

7.7 柱状图组件、饼形图组件和曲线图组件

7.7.1 柱状图组件、饼形图组件和曲线图组件概述

柱状图组件、饼形图组件和曲线图组件可以通过一定比例的柱形、饼形和曲线来显示一定数量的数据,将数据图形化,使其看起来更加直观。例如,将几年的销售额用曲线图表示,可以直观地看出销售额的增长和减少状况。

这三个控件都是只有属性,没有专有方法和自有事件,并且属性都类似。所以本节将主要介绍柱状图组件的属性及使用,其他两个组件将不再详细介绍。

7.7.2 柱状图组件、饼形图组件和曲线图组件属性

1. “有无边框”、“边框颜色”属性

这两个属性设置柱状图组件是否有边框及边框的颜色。如图7-55所示。

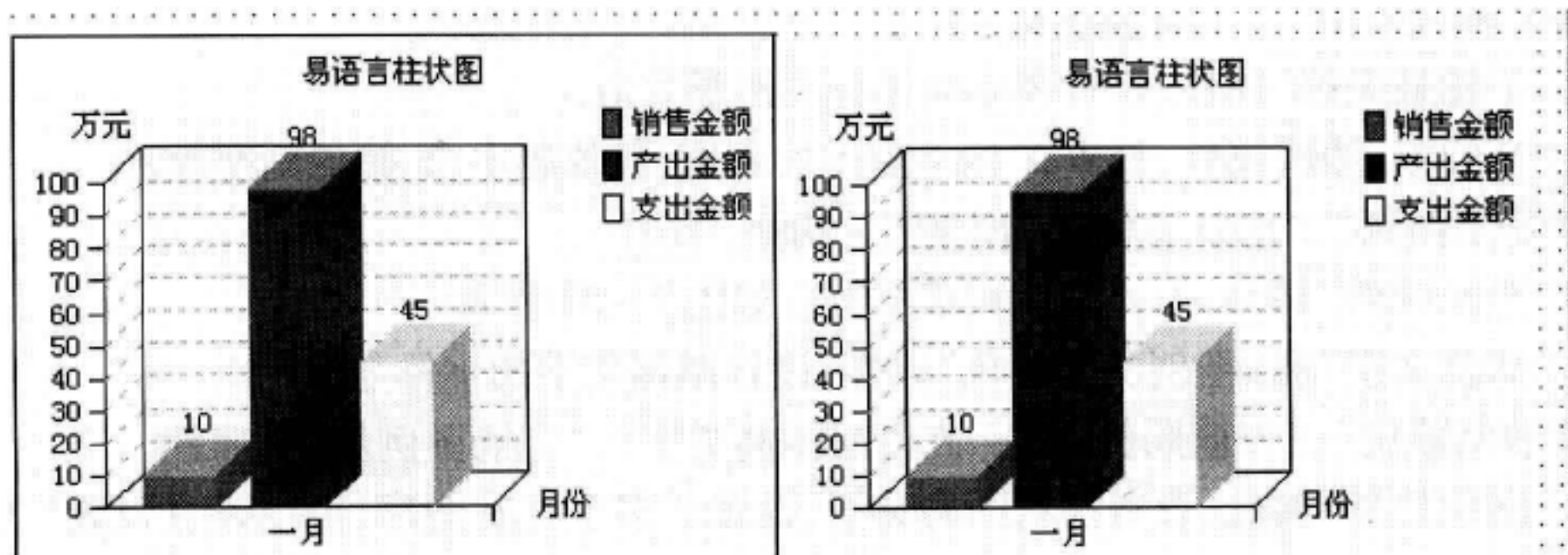


图7-55 有边框(左)及无边框(右)的柱状图控件

2. “背景颜色”、“背景图片”、“拉伸底图”属性

- “背景颜色”属性,设置柱状图组件的背景颜色。
- “背景图片”属性,设置柱状图组件的背景图片。
- “拉伸底图”属性,若设置为“真”,则组件的底图将随组件的拉伸而改变大小。

3. “三维图形”属性

该属性用于设置柱状图是否具有三维的显示效果。如图 7-56 所示。

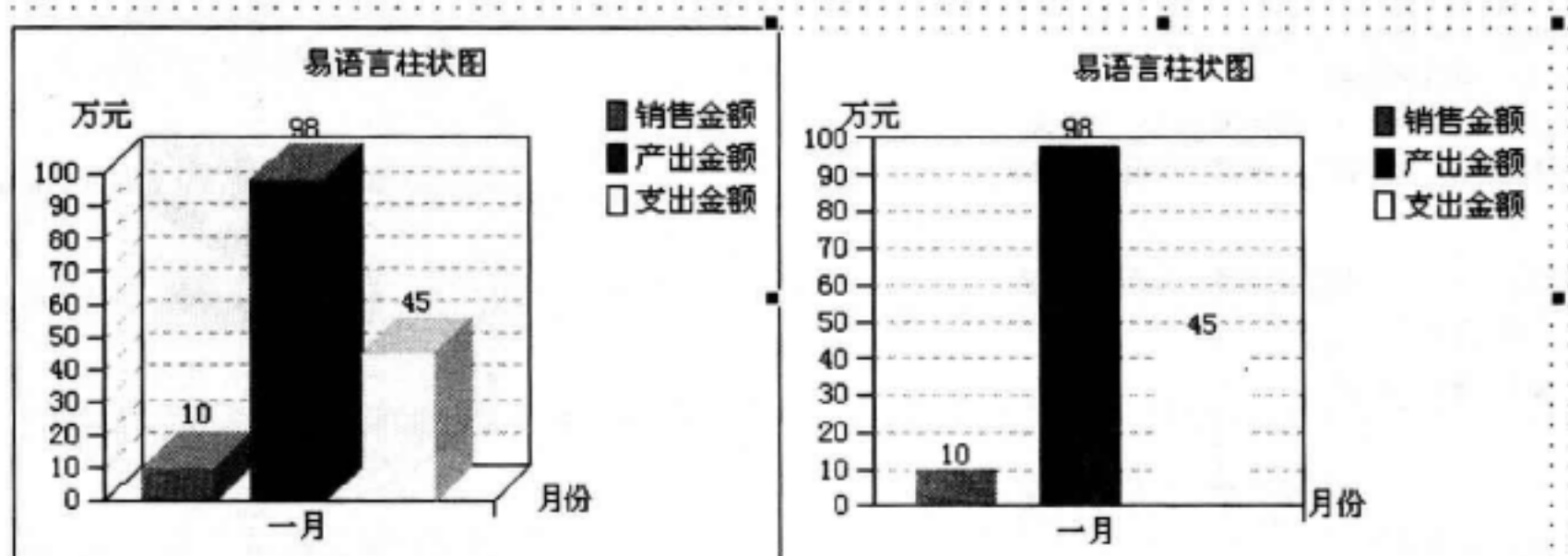


图 7-56 柱状图三维(左)效果和二维效果(右)

4. “显示标题”、“标题文字”、“标题字体”和“标题颜色”属性

• “显示标题”属性,设置是否显示柱状图标题,柱状图的标题为“易语言柱状图”,这也是该组件的默认标题。

- “标题文字”属性,设置柱状图的标题文字。
- “标题字体”和“标题颜色”属性,设置柱状图标题文字的字体和字体颜色。

5. “X 轴单位”、“Y 轴单位”属性

• “X 轴单位”属性为显示在 X 轴(横坐标轴)右端的单位文字,如图 7-57 中在横轴最右边显示的“月份”是该柱状图的 X 轴单位。

• “Y 轴单位”属性为显示在 Y 轴(纵坐标轴)顶端的单位文字,如图 7-57 中在纵轴最上边显示的“万元”是该柱状图的 Y 轴单位。

程序中,根据实际情况,来设定 X 轴和 Y 轴的单位。

6. “标注字体”、“标注颜色”属性

用于设置组件中所有横轴和纵轴标注文字的字体和文字颜色。标注包括组件的“X 轴单位”、“Y 轴单位”、“当前 X 轴标注文字”、“Y 轴上刻度上的数字”、“柱状图顶端的数字”。

7. “Y 轴最大值”、“Y 轴刻度数”属性

- “Y 轴最大值”属性,设置纵坐标轴标注的最大值。
- “Y 轴刻度数”属性,设置 Y 轴上刻度的密度,数值越小,Y 轴的刻度越密。

8. “显示图例”、“图例字体”、“图例文字颜色”属性

- “显示图例”属性,设置是否显示图例。如图 7-57 所示。
- “图例文字”和“图例文字颜色”属性,设置图例文字的字体与颜色。

9. “图例数量”、“当前图例索引”、“当前图例文字”、“当前图例颜色”属性

• “图例数量”属性,设置柱状图的数量,设置了多个柱状图后,可以通过设置“当前图例索引”属性,来对不同的柱状图进行设置。

- “当前图例索引”,当前控制的图例索引,从 0 开始,即 0 为第一个图例,1 为第二个图

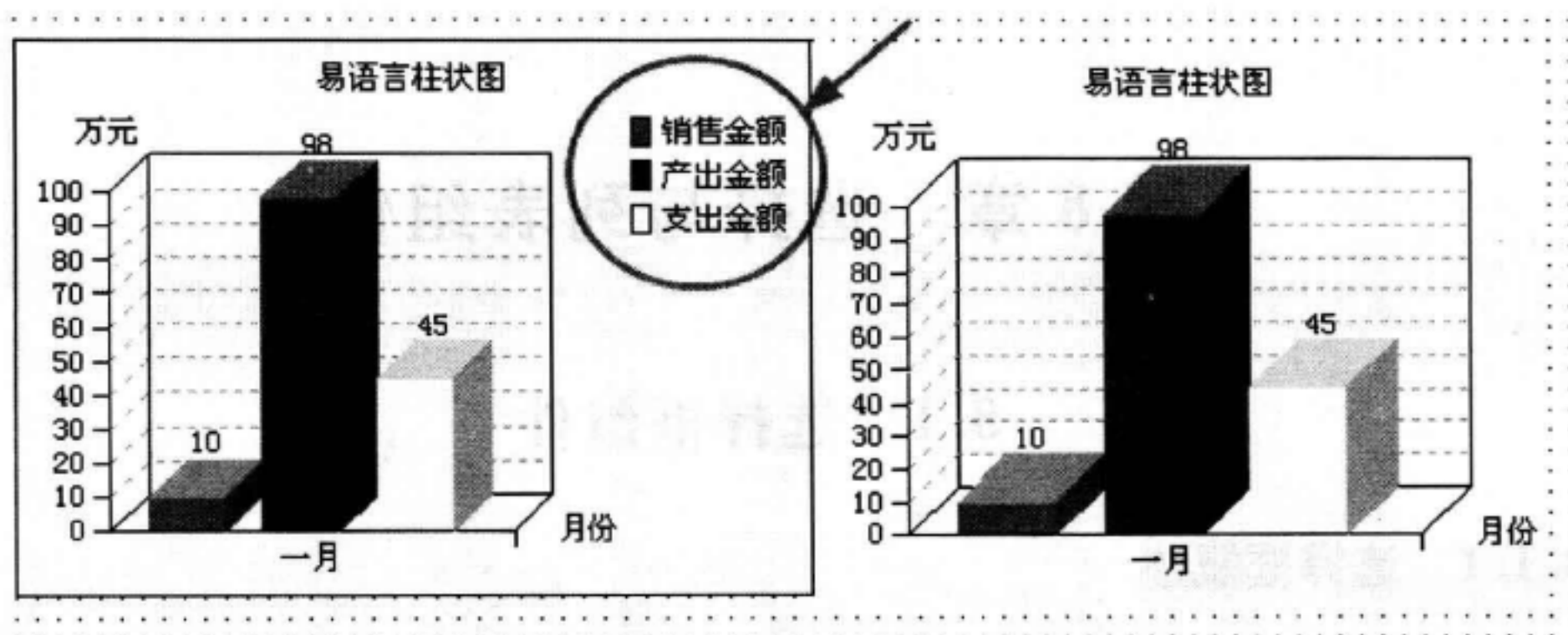


图 7-57 显示图例(左)与不显示图例(右)的柱状图组件

例,依次类推。

- “当前图例文字”属性,设置当前控制的图例的文字,即显示在组件右上角的图例提示。
- “当前图例颜色”属性,设置当前柱状图的颜色。

10. “X 轴标注数量”、“当前 X 轴标注索引”、“当前 X 轴标注文字”属性

• “X 轴标注数量”属性,设置横坐标中标注的数量,设置多个标注后,可以通过改变“当前 X 轴标注索引”来对不同的标注进行设置。

- “当前 X 轴标注索引”属性,设置当前控制的横坐标标注。
- “当前 X 标注文字”是横轴下方的文字。

11. “当前数据值”、“显示小数位”属性

• “当前数据值”属性,设置当前柱状图所显示的数值,即表示当前柱状图的高度,该数值在柱状图的顶部显示。

- “显示小数位”属性为显示在柱状图顶部的数值显示的小数位数。

第 8 章 选择与列表组件

8.1 选择框组件

8.1.1 选择框概述

选择框 ☐ 又称“复选框”，该组件提供一组输入项目（“是/否”或者“真/假”选项），用户可以选择其中的一项或多项。当选定某选择框时，该组件左边的小方框内将以“√”符号标记。如图 8-1 所示。

如果在同一窗口中有多组这样的输入项目，可以使用包容型窗口单元（容器控件）（如分组框或图片框）把它们分组，同一组的选择框放置在同一个容器控件中。如图 8-2 所示。

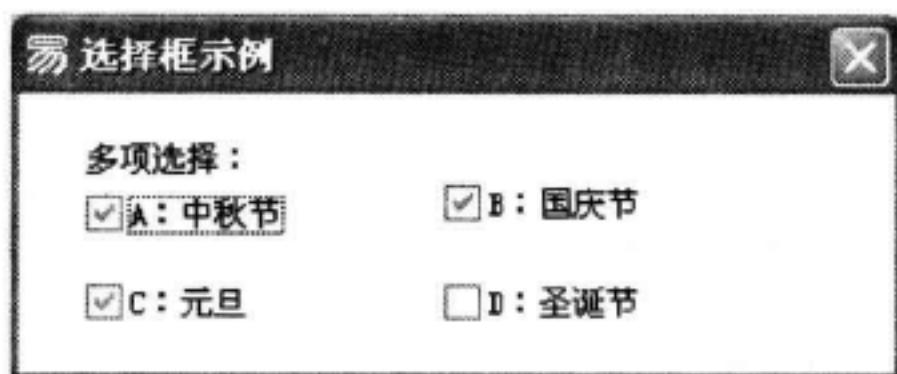


图 8-1 选择框示例

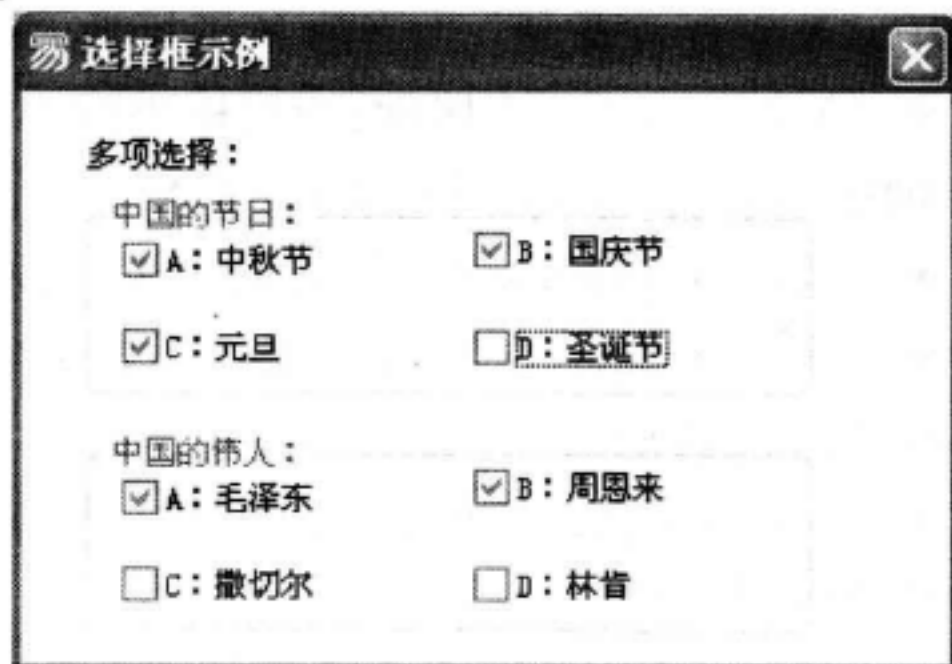


图 8-2 多组选择框示例

选择框没有自己的方法，下面重点介绍选择框的属性和事件。

8.1.2 选择框的属性

选择框组件的属性主要有“图片”、“标题”、“选中”、“标题居左”、“按钮形式”、“数据源”、“数据列”等，此外还有一些其他属性可以加强选择框的技巧应用，如“平面”、“横向对齐方式”、“纵向对齐方式”、“文本颜色”、“背景颜色”、“字体”等。在程序中既可以设置各种属性的值，也可以读取各种属性的值。

选择框组件通常只用到“标题”和“选中”这两个属性。选择框的“标题”属性用于显示选择框中的文字，作为选项的介绍或提示。“选中”属性内容为逻辑型，只能为“真”或“假”，默认为“假”。如果“选中”属性为“真”，则选择框中打“√”号。

在编程中，可以默认一个选择框为非选中状态，也可以默认为选中状态，只需要对“选中”属性的值进行相应设置就行了。参看例程 8-1。

选择框组件的“图片”属性可以让选择框的提示以图片形式表示，这样选择框的显示更加丰富。在使用“图片”属性设置时，选择框“标题”属性内容不再显示。另外，图片属性中的图片一般需要自行绘制，图 8-3 所示就是一个绘制好的图片。

点击属性“图片”右侧的按钮,在弹出的“图片属性对话框”中更换图片文件“撒切尔.jpg”,效果如图8-4所示,可以看到原来选择框中的标题“撒切尔”文字已经被“撒切尔”图片替换。

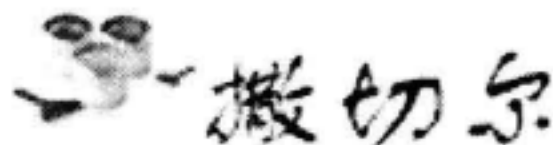


图 8-3 图片选项“撒切尔”

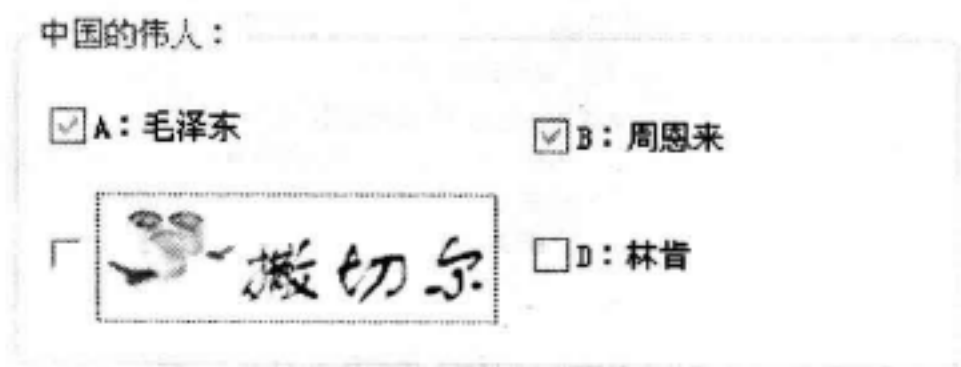


图 8-4 选择框“图片”属性应用示例

选择框“标题居左”属性一般为“假”,即通常的显示方式,如果该属性设置为“真”,则选择框标题在“☐”的左边显示,如图8-5所示。

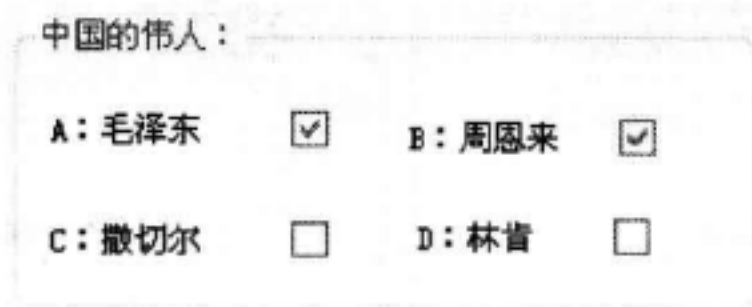


图 8-5 选择框“标题居左”属性应用示例

选择框属性“按钮形式”一般为假,当该值设置为真时,选择框以按钮的形式在界面上表现,图8-6所示为属性“按钮形式”为真时,选择框设计时和运行时的形态。

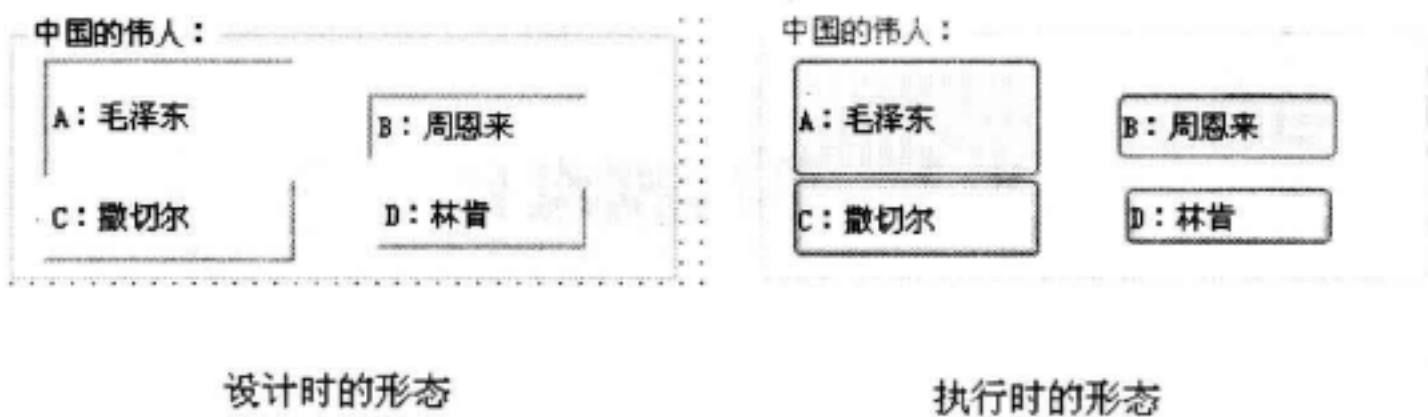


图 8-6 选择框“按钮形式”属性应用示例

前面讲到的选择框的几个属性示例,具体参考例程8-1。
选择框还有两个重要的属性:“数据源”和“数据列”,这样就可以直接从易语言数据库中对选择框“标题”进行数据更新,由于这两个属性的使用要用到“数据源”、“通用提供者”或“数据库提供者”等属性,控制较为复杂,本书作为入门手册,限于篇幅,就不做更多介绍,待以后的高级应用中再做讲解。

8.1.3 选择框的事件

选择框组件的事件只有一个,即“被单击”。当选择框被单击时会产生此事件。要注意的是,本事件只是对选择框被单击这个动作作出反应,并不判断选择框的“选中”状态,因此,在事件处理中,要对选择框的“选中”状态进行判断,并作出处理。下面通过一个简单的例子认识其功能。

【范例8-1】使用选择框来控制一个标签控件的显示状态。具体参考例程8-2。

“_选择框_隐藏标题_被单击”子程序在单击事件发生后,先查看选择框的状态,如果“选中”属性为真,则询问是否真的隐藏标签,如果确认隐藏,则设置标签“可视”属性为假。如果“选中”属性为假,则标签不隐藏,设置标签“可视”属性为真。如图 8-7 所示。

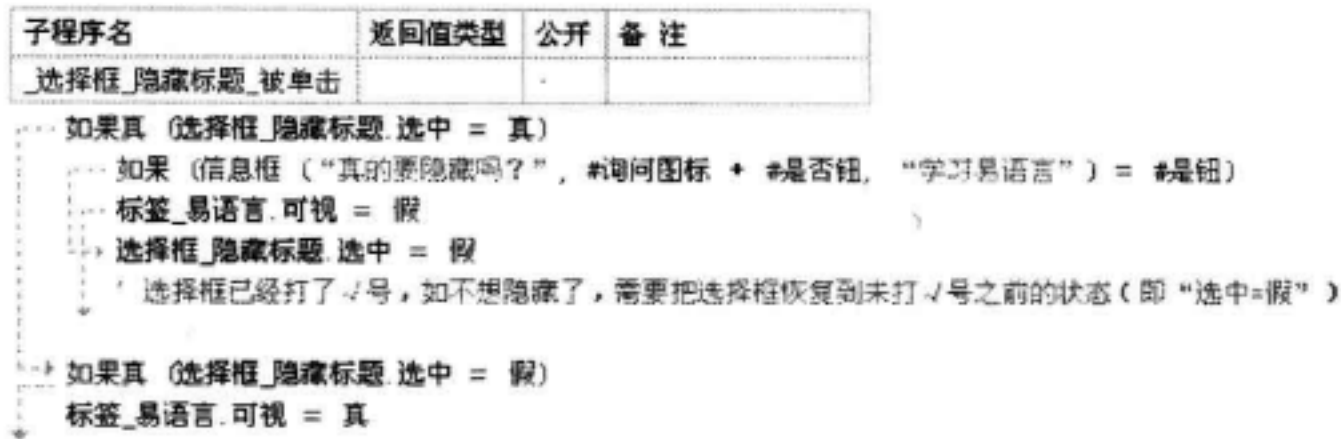


图 8-7 “选择框_被单击”代码示例

【运行结果】运行例程 8-2,观测单击选择框前后标签的变化,如图 8-8 所示。

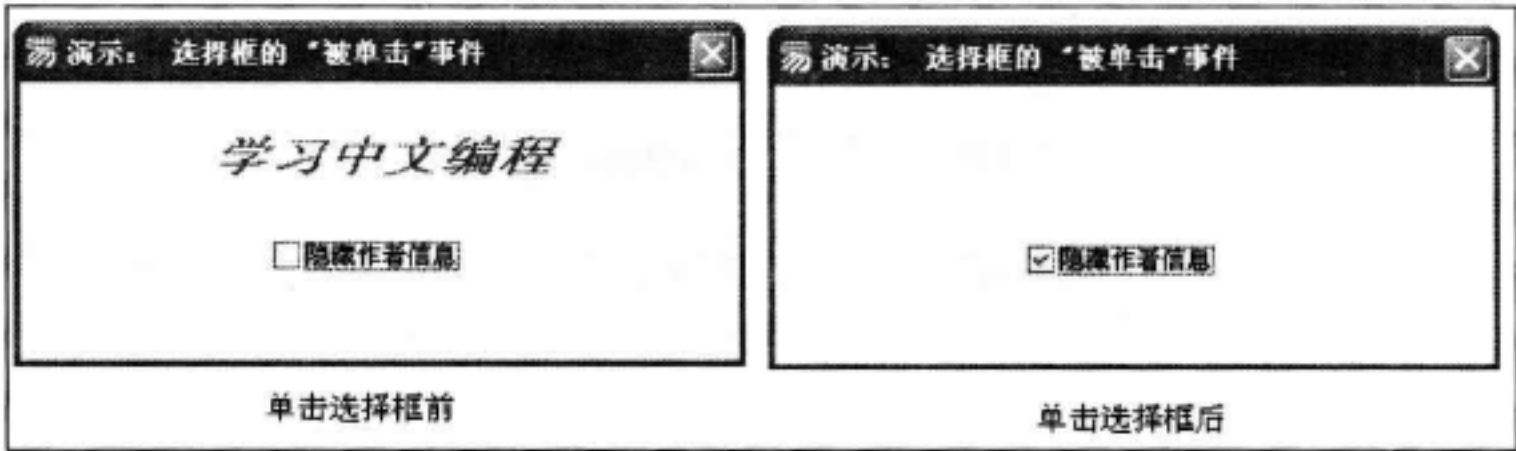



图 8-8 “选择框_被单击”运行结果示例

【注意】单击选择框时,会产生触发事件子程序,在事件子程序再对选择框的“选中”属性进行判断。

8.2 单选框组件

8.2.1 单选框概述

单选框 ,该组件的特点是同一组单选框中,只能选择其中之一。提供一组输入项目 (“是/否”或者“真/假”选项),用户只能选择其中的一项。当选定单选框时,该组件左边的小圆框内将以“·”符号标记。如图 8-9 所示,具体参考例程 8-3。

如果在同一窗口中有多组这样的输入项目,可以使用包容型窗口单元(容器控件)(如分组框或图片框)把它们分组,同一组的单选框放置在同一个容器控件中。如图 8-10 所示,具体参考例程 8-4。

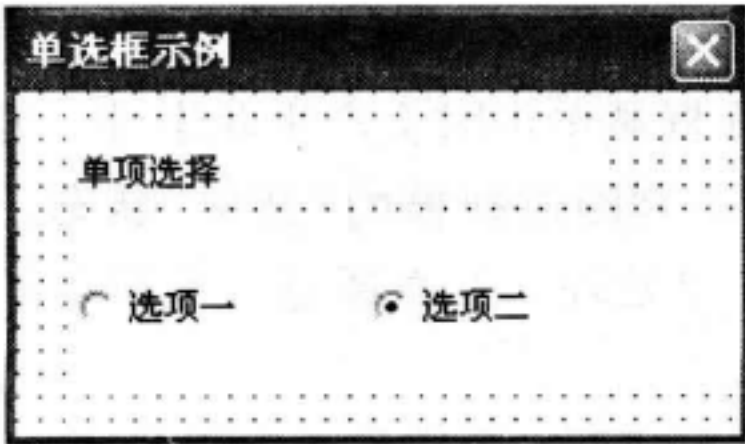


图 8-9 单选框示例



图 8-10 多组单选框示例

单选框没有自己的方法,下面重点介绍单选框的属性和事件。

8.2.2 单选框的属性

单选框组件的属性主要有“图片”、“标题”、“选中”、“标题居左”、“按钮形式”、“数据源”、“数据列”等,此外还有一些其他属性可以加强选择框的技巧应用,如“平面”、“横向对齐方式”、“纵向对齐方式”、“文本颜色”、“背景颜色”、“字体”等。在程序中既可以设置各种属性的值,也可以读取各种属性的值。

单选框组件通常只用到“标题”和“选中”这两个属性。单选框的“标题”属性用于显示单选框中的文字,作为选项的介绍或提示。“选中”属性内容为逻辑型,只能为“真”或“假”,默认为“假”。如果“选中”属性为“真”,则选择框中打“·”号。

在编程中,可以默认一个单选框为非选中状态,也可以默认为选中状态,只需要对“选中”属性的值进行相应设置就行了。参看例程8-4。

单选框组件的“图片”属性可以让单选框的提示以图片形式表示,这样单选框的显示更加丰富。在使用“图片”属性设置时,单选框“标题”属性内容不再显示。另外,图片属性中的图片一般需要自行绘制,图8-11所示就是一个绘制好的图片。



图 8-11 图片选项
“撒切尔”

点击属性“图片”右侧的按钮,在弹出的“图片属性对话框”中更换图片文件“撒切尔.jpg”,效果如图8-12所示,可以看到原来单选框中的标题“撒切尔”文字已经被“撒切尔”图片替换。

单选框“标题居左”属性一般为“假”,即通常的显示方式,如果该属性设置为“真”,则单选框标题在“·”的左边显示,如图8-13所示。

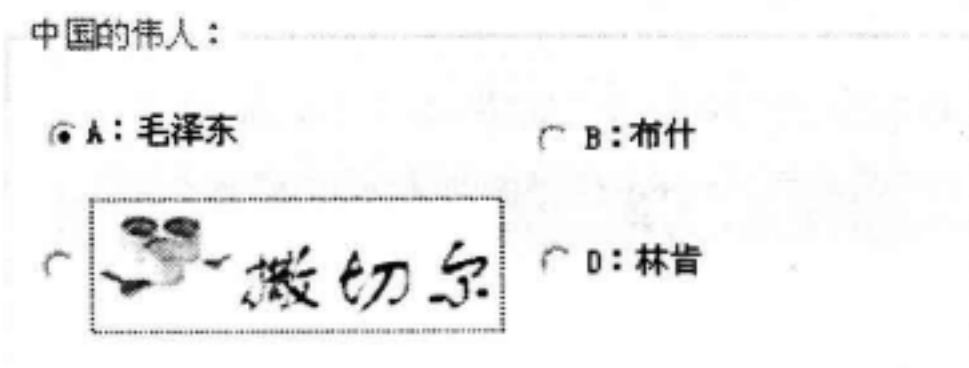


图 8-12 单选框“图片”属性应用示例

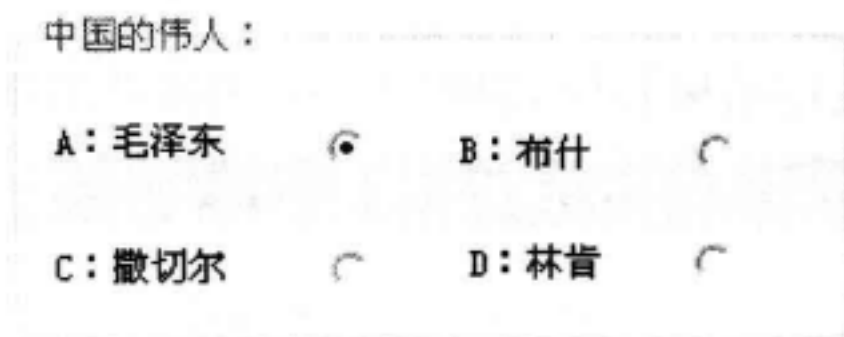


图 8-13 单选框“标题居左”属性应用示例

单选框属性“按钮形式”一般为假,当该值设置为真的时,单选框以按钮的形式在界面上表现,如图8-14所示为属性“按钮形式”为真时,单选框设计时和运行时的形态。具体参考例程8-5。

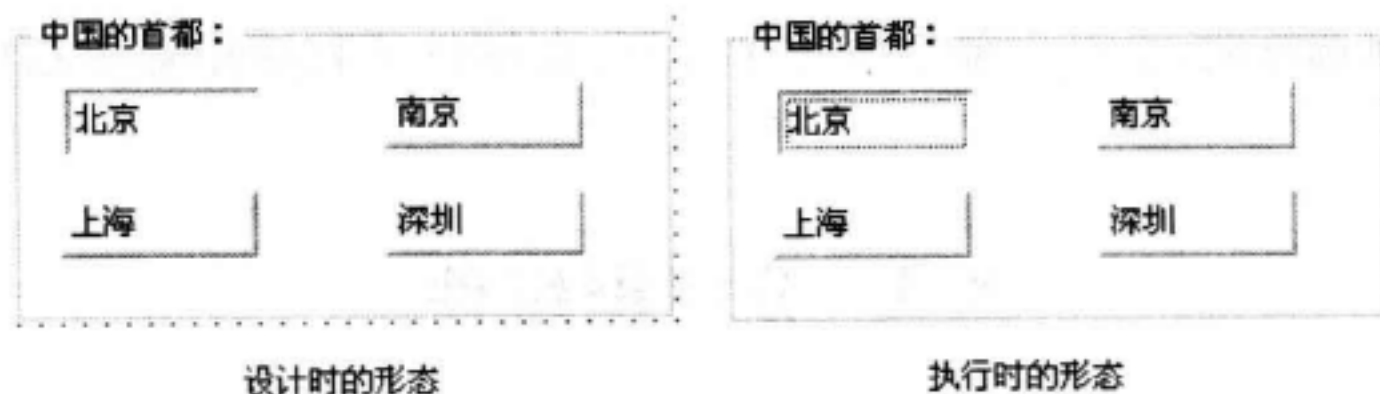


图 8-14 单选框“按钮形式”属性应用示例

单选框还有两个重要的属性:“数据源”和“数据列”,这样就可以直接从易语言数据库中对单选框“标题”进行数据更新,由于这两个属性的使用要用到“数据源”、“通用提供者”,或

“数据库提供者”等属性,控制较为复杂,本书作为入门手册,限于篇幅,就不做更多介绍,待以后的高级应用中再做讲解。

8.2.3 单选框的事件

单选框组件的事件只有一个,即“被单击”。当单选框被单击时会产生此事件。要注意的是,本事件只是对单选框被单击这个动作作出反应,并不判断选择框的“选中”状态,因此,在事件处理中,要对单选框的“选中”状态进行判断,并作出处理。下面通过一个简单的例子认识其功能。

【范例 8-2】使用单选框来控制一个标签控件的显示状态。具体参考例程 8-6。

“_单选框_隐藏标题_被单击”子程序在单击事件发生后,先查看单选框的状态,如果“选中”属性为真,则询问是否真的隐藏标签,如果确认隐藏,则设置标签“可视”属性为假。如果“选中”属性为假,则标签不隐藏,设置标签“可视”属性为真。具体的程序代码如图 8-15 所示。

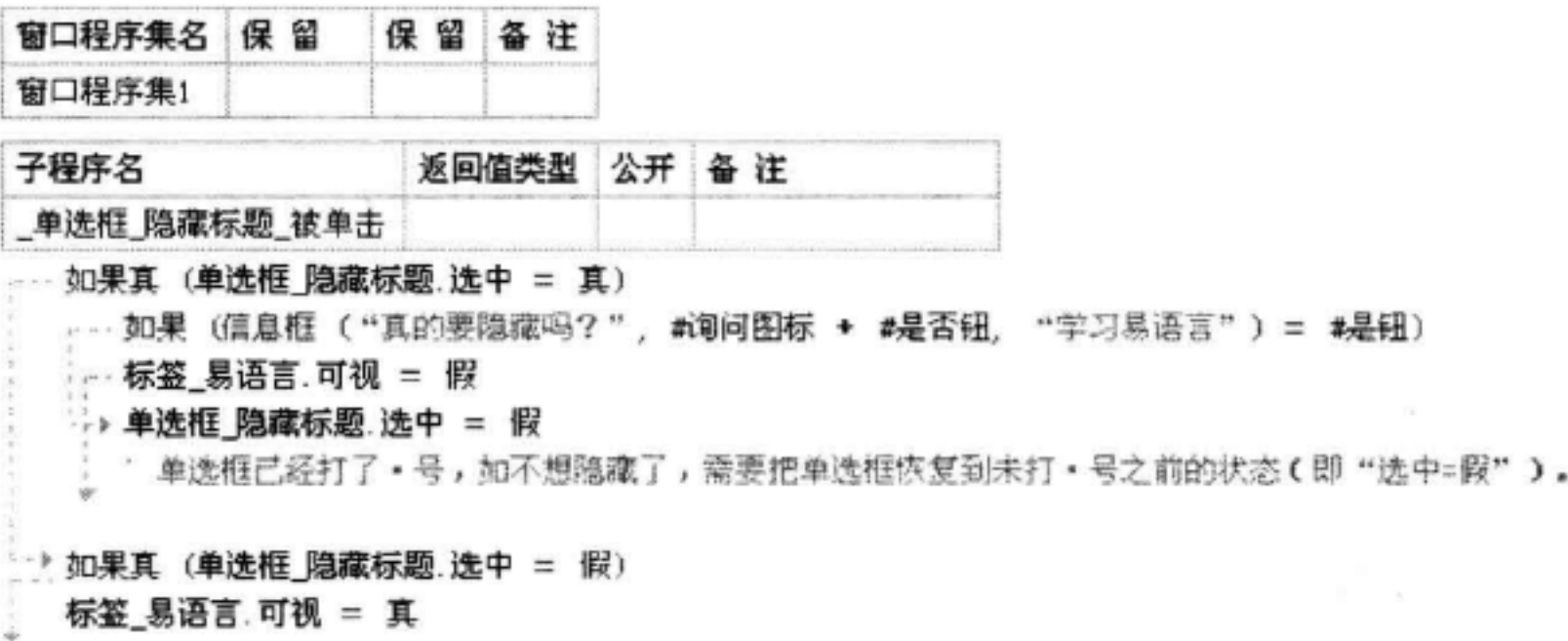


图 8-15 “单选框_被单击”代码示例

【运行结果】运行例程 8-6,观测单击单选框前后标签的变化,如图 8-16 所示。

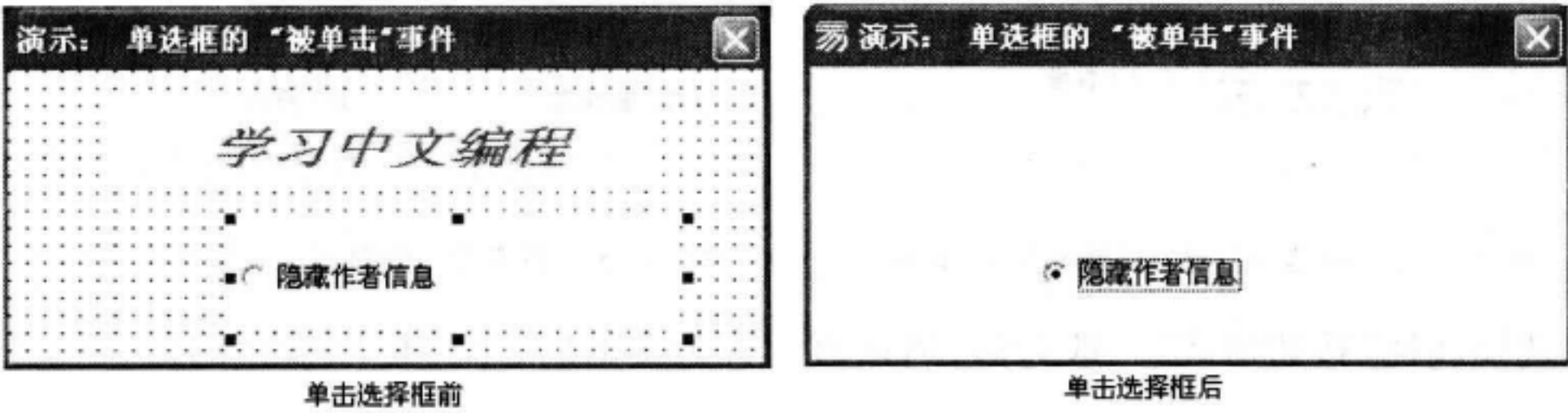



图 8-16 “单选框_被单击”运行结果示例

【注意】单击单选框时,会产生触发事件子程序,在事件子程序中对单选框的“选中”属性进行判断。

8.3 分组框组件

8.3.1 分组框概述

分组框组件 ,该组件最大的作用就是充当选择框、单选框的父组件,用于形成不同的分组。当分组框组件主要作为选择框、单选框的父组件,通常只需设置其“标题”属性。在本书

8.1 和 8.2 节中已经介绍了分组框与选择框和单选框的分组用法,这里就不再赘述。

分组框没有自己的方法,下面重点介绍分组框的属性和事件。

8.3.2 分组框的属性

分组框组件的主要属性有“标题”、“对齐方式”、“文本颜色”、“背景颜色”、“字体”等。“标题”属性用于显示分组框的分组名称,根据“对齐方式”属性的文字决定标题在分组框中的位置,可以显示于分组框组件的左上角、右上角或上面中间。“对齐方式”属性的内容为整数型,有3个可选值:“0. 左边”、“1. 居中”、“2. 右边”,通常默认值为“0. 左边”,如图8-17所示,具体参考例程8-7。

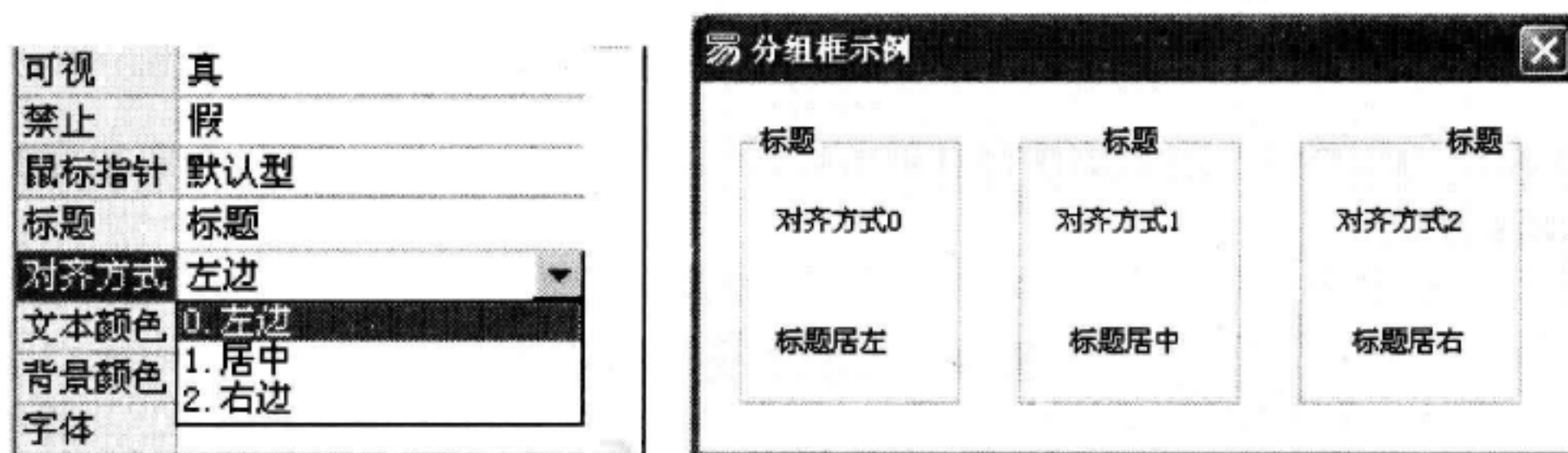


图 8-17 分组框示例

8.3.3 分组框的事件

分组框的事件有“鼠标左键被按下”、“鼠标左键被放开”、“鼠标右键被按下”、“鼠标右键被放开”、“被双击”、“鼠标位置被移动”。对于分组框来说,它的基本事件几乎不响应。下面通过一个简单的例子认识其功能。

【范例8-3】利用六个选择框来验证分组框的事件。具体参考例程8-8。

新建一个易程序,在窗口中添加六个选择框组件和一个分组框组件,分组框组件的鼠标左键被按下事件和鼠标左键被放开事件被触发的代码如图8-18所示。

窗口程序集名	保 留	保 留	备 注			
窗口程序集1						

子程序名	返回值类型	公开	备 注			
_分组框1_鼠标左键被按下	逻辑型					
参数名	类 型	参考	可空	数组	备 注	
横向位置	整数型					
纵向位置	整数型					
功能键状态	整数型					

--- 如果真 (选择框1.选中 = 真)
 _启动窗口.标题 = “鼠标左键被按下”

子程序名	返回值类型	公开	备 注			
_分组框1_鼠标左键被放开	逻辑型					
参数名	类 型	参考	可空	数组	备 注	
横向位置	整数型					
纵向位置	整数型					
功能键状态	整数型					

--- 如果真 (选择框2.选中 = 真)
 _启动窗口.标题 = “鼠标左键被放开”

图 8-18 分组框事件代码示例

【运行结果】运行例程 8-8, 观测单击单选框前后的变化, 如图 8-19 所示。

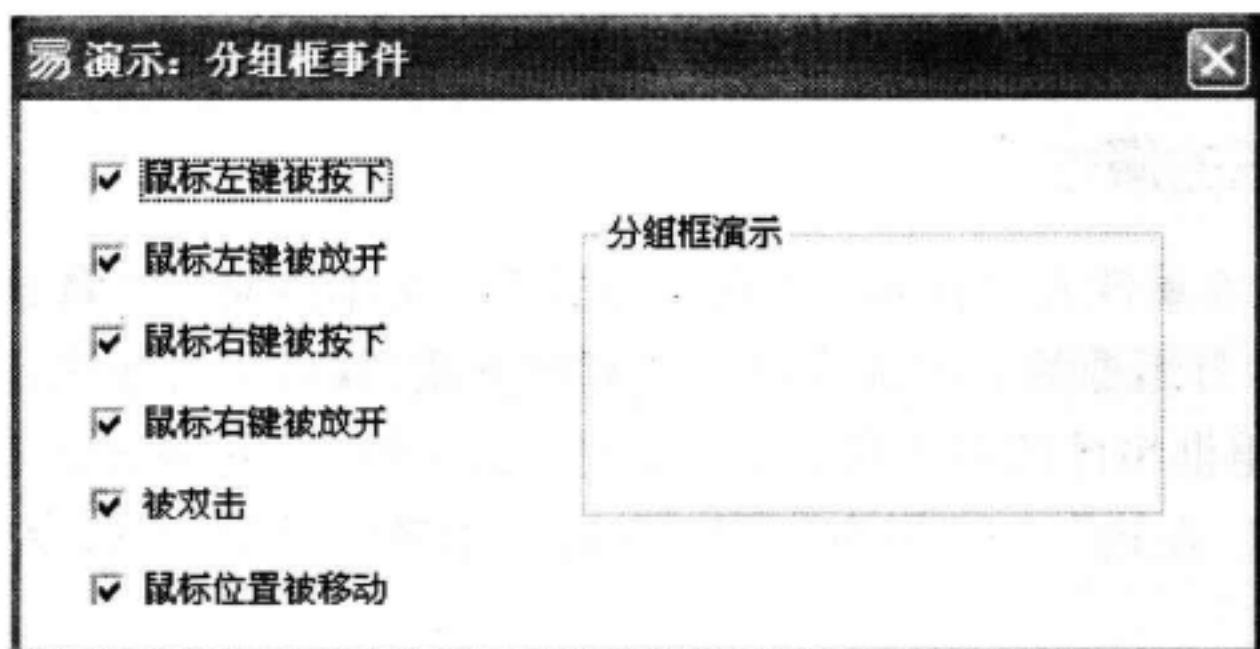


图 8-19 分组框事件运行结果示例

【注意】一般分组框组件没有响应任何事件, 但是不影响它的分组功能, 由它包容的其他组件响应事件。

8.4 选择夹组件

8.4.1 选择夹概述

选择夹是一个重要的组件, 通常作为一种包容型窗口单元(容器型组件)。一个选择夹组件可以拥有多个子夹, 每一个子夹中都可以单独摆放其他组件, 各子夹互不影响。利用选择夹组件可以在相对较小的空间内摆放相对较多的组件, 从而为充分利用空间提供了一套可选方案。

选择夹通过各子夹的浮动条可以快速地切换到各子夹的界面, 其中一个子夹就相当于一个窗口。如果使用弹出多个窗口, 有的变量就要重新定义, 不太方便。使用选择夹就不用考虑太多的变量传递, 从而简化了程序集中变量的定义。可见, 选择夹比窗口间的切换更方便, 在有可能的情况下, 要多用选择夹, 少用弹出窗口, 如图 8-20 所示, 具体参考例程 8-9。

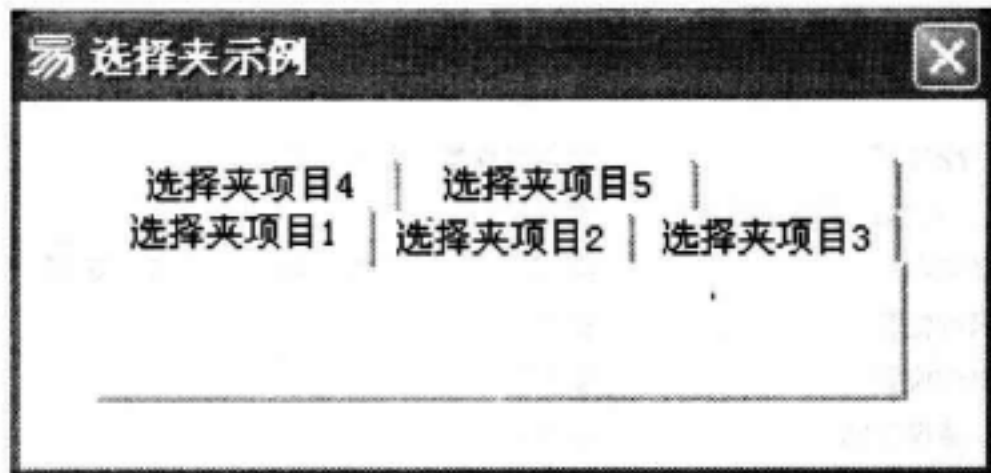


图 8-20 选择夹示例

下面重点介绍选择夹的属性、方法和事件。

8.4.2 选择夹的属性

选择夹的重要属性有“子夹管理”、“现行子夹”、“表头方向”、“允许多行表头”、“隐藏自身”等。

使用选择夹时, 必须设置“子夹管理”属性, 它仅用于设计期, 负责提供一个“子夹管理对话框”, 以设置选择夹的各子夹。“子夹管理”设置的步骤是, 在弹出“子夹管理对话框”中分别

设置各个子夹的名称,确定后返回。图8-21所示,为选择夹设置了5个子夹,各子夹名称分别为“选择夹项目1”、“选择夹项目2”、“选择夹项目3”、“选择夹项目4”和“选择夹项目5”等。

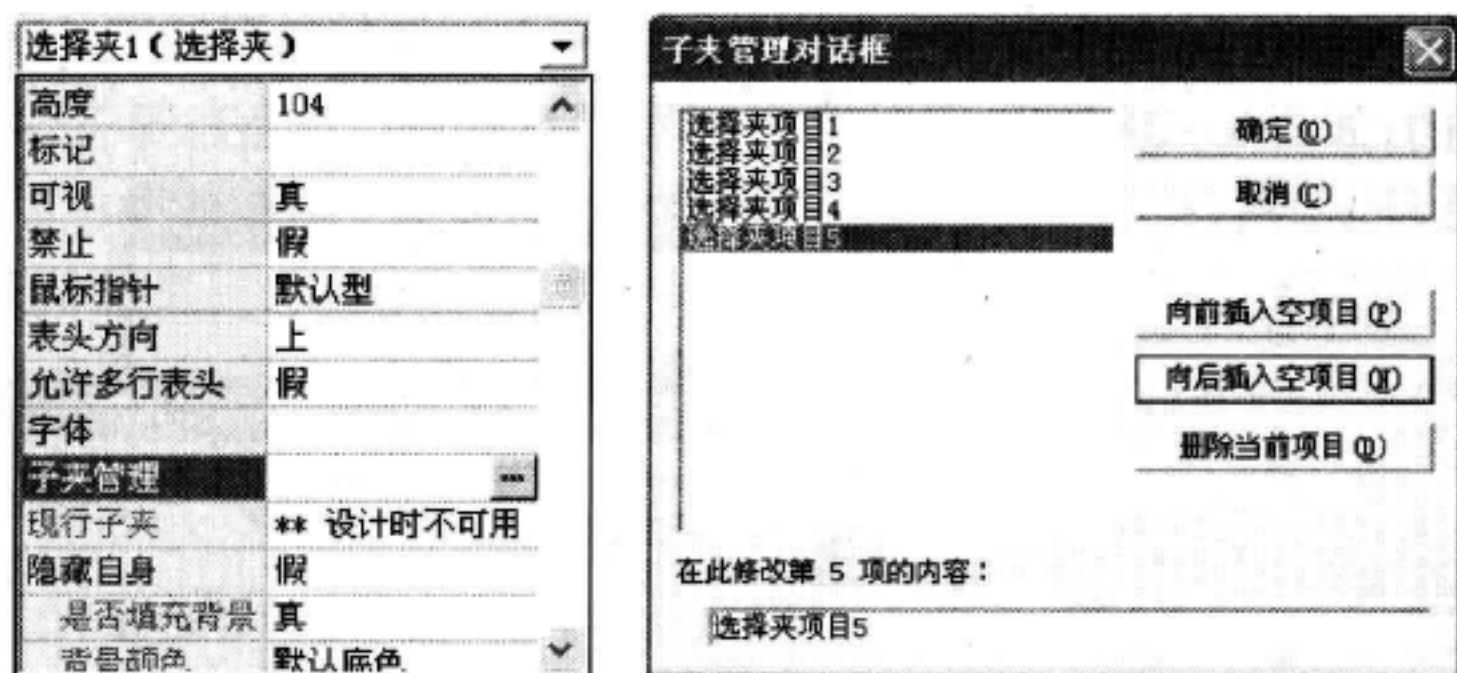


图8-21 选择夹“子夹管理对话框”

选择夹的“现行子夹”属性在设计期无效,仅用于运行期。内容为整数型,用于指定现行“被选中”子夹的索引,就是各子夹的编号,从0开始,即第一个子夹的索引是0,依次递增。在程序中既可以读取该属性,也可以为该属性赋值(为其赋值相当于切换子夹),如图8-22所示,具体参考例程8-10。

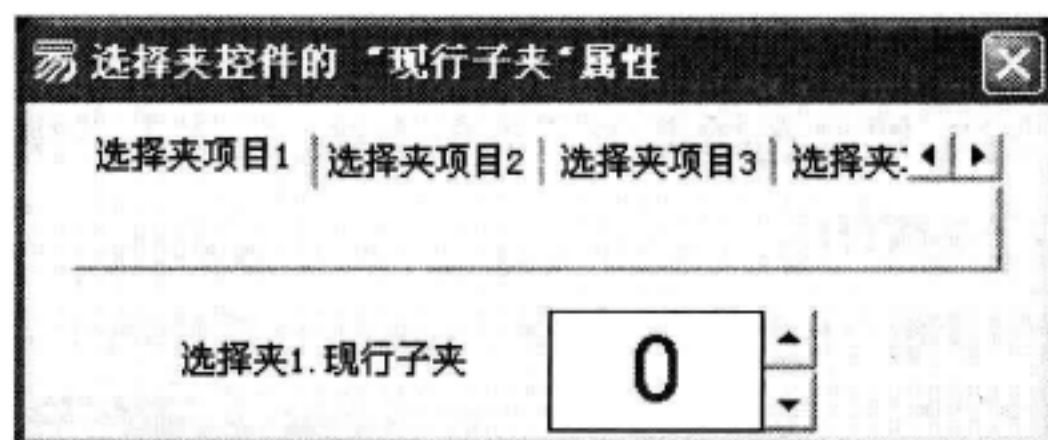


图8-22 选择夹“现行子夹”

“表头方向”属性内容为整数型,有4个可选值:“0. 上”、“1. 下”、“2. 左”、“3. 右”,平常见到的选择夹,默认为“0. 上”,表头方向向上。在图8-23所示的窗口中,4个选择夹组件的表头方向分别为左、上、下、右。具体参考例程8-11。

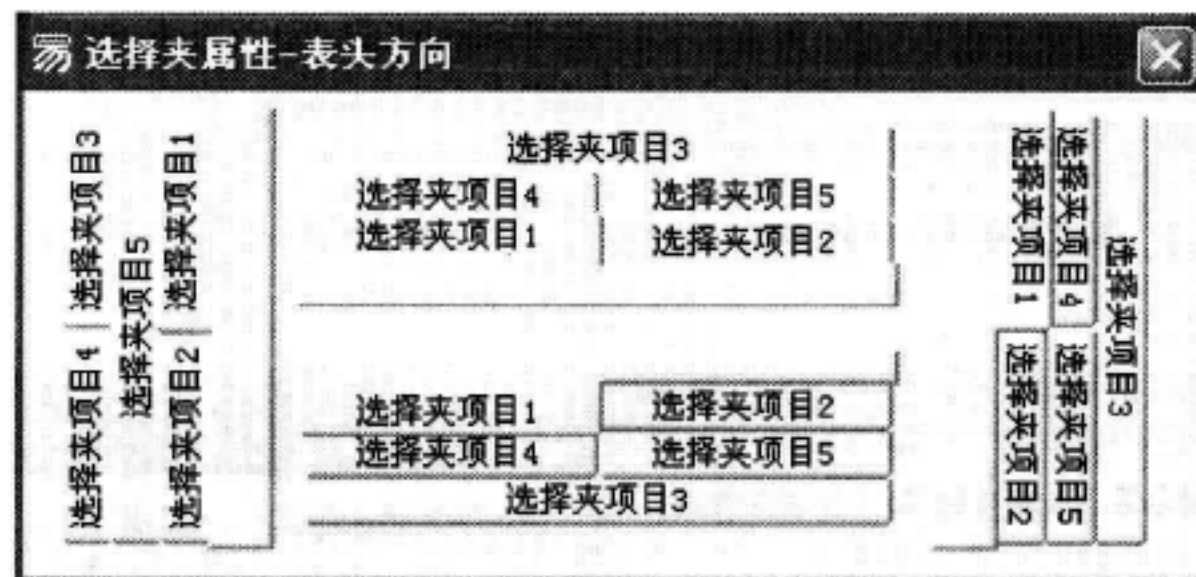


图8-23 选择夹“表头方向”

“允许多行表头”属性控制表头是否以多行显示,内容为逻辑型,只能为“真”或“假”,默认为“假”,即单行显示,而不以多行显示。如果单行显示时宽度不够无法显示出所有子夹,则右上角(或右下角)出现 ◀ || ▶,如前图8-22所示。

“隐藏自身”属性,控制是否隐藏选择夹自身,内容为逻辑型,只能为“真”或“假”,默认为

“假”。所谓“隐藏自身”，是指在程序运行时只能看到选择夹中的其他组件，却不能看到选择夹自身。利用此属性可使窗口上放置许多组件，但并不显拥挤。

需要注意的是，一旦设置隐藏自身属性为“真”后，用户看不到选择夹，因此不能通过单击表头的方法在各子夹间切换，而只能通过程序员给选择夹的现行子夹属性编码赋值的方法，达到切换子夹的目的，如图 8-24 所示，具体参考例程 8-12。

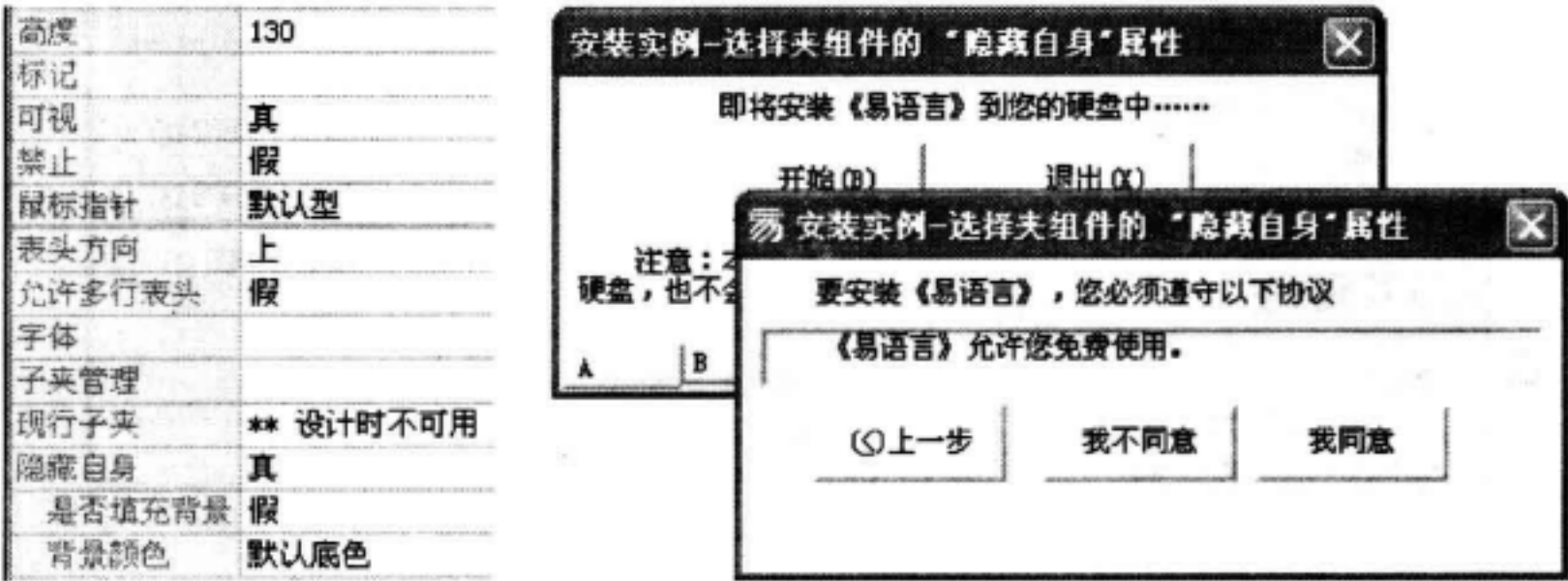


图 8-24 选择夹“隐藏自身”

8.4.3 选择夹的方法

选择夹的重要方法有“取子夹数目()”、“取子夹名称()”、“置子夹名称()”，均存在于支持库面板中。

1. “取子夹数目()”方法，即取选择夹中子夹的个数，具体参考例程 8-13。

子夹数 = 选择夹 1. 取子夹数目()

上述代码执行后，整数型变量“子夹数”中保存了“选择夹 1”中的子夹个数，如图 8-25 所示。

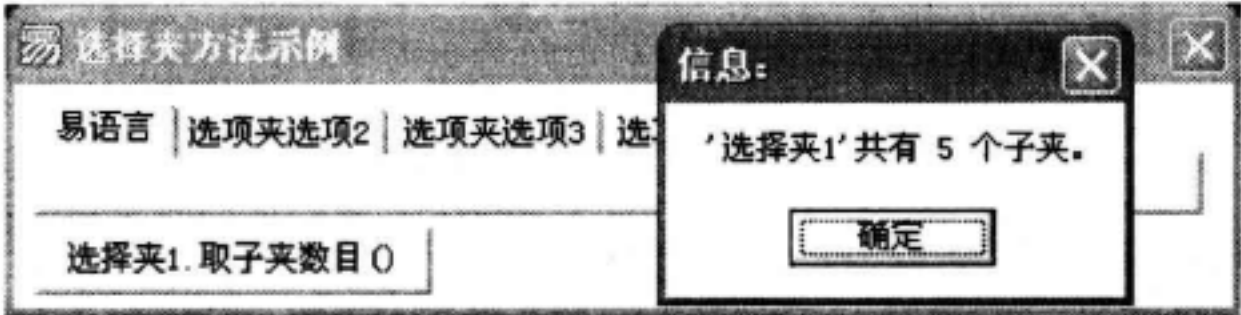


图 8-25 选择夹“取子夹数目”

2. “取子夹名称()”方法，即取选择夹中指定子夹的名称，具体参考例程 8-14。

名称 = 选择夹 1. 取子夹名称(1)

上述代码执行后，文本型变量“名称”中保存了“选择夹 1”中第一个子夹的名称。如图 8-26 所示。

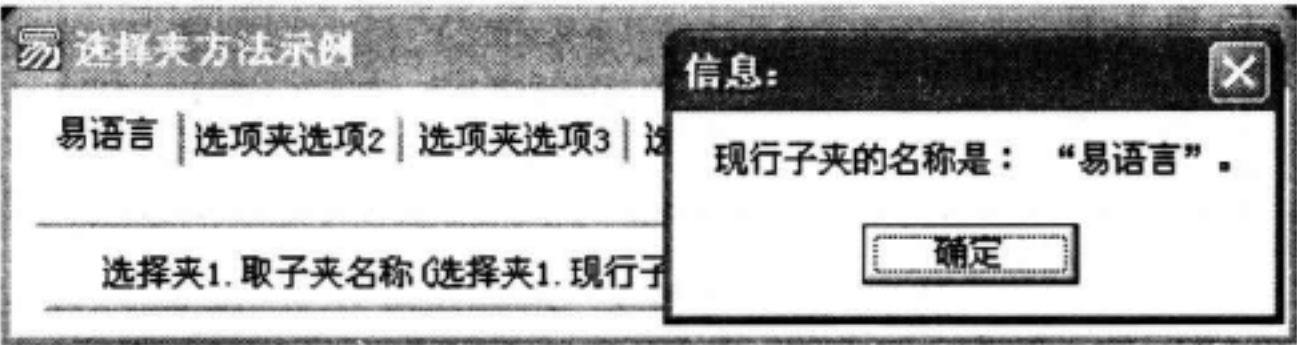


图 8-26 选择夹“取子夹名称”

3. “置子夹名称()”方法，即设置选择夹中指定子夹的名称，具体参考例程 8-15。

选择夹 1. 置子夹名称(1, “新选项夹选项 1”)

上述代码执行后,“选择夹 1”中第一个子夹的名称“易语言”被改为“新选项夹选项 1”,如图 8-27 所示。

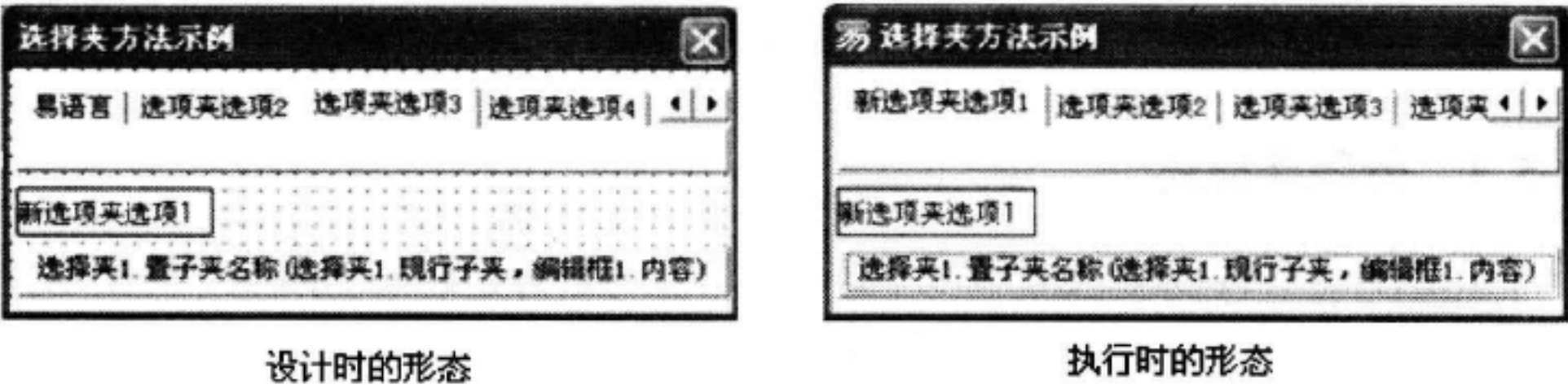


图 8-27 选择夹“置子夹名称”

8.4.4 选择夹的事件

选择夹的重要事件有“子夹被改变”和“将改变子夹”。“子夹被改变”事件是在现行被选择子夹改变时产生。在该事件的处理子程序中读取现行子夹属性,即得当前被选择子夹。“将改变子夹”事件是在现行被选择子夹即将被改变时产生。该事件有一个逻辑型返回值。如果在本事件的处理子程序中返回假,表示不允许改变子夹;返回真或不返回值则允许改变。下面通过一个简单的例子认识其功能。

选择夹的两个事件也不大常用,初学者可以跳过。

【范例 8-4】在选择夹中有多个项目,只有选中规定的内容才能切换到其他子夹。具体参考例程 8-16。

“_选择夹 1_将改变子夹”子程序在现行被选择子夹改变事件发生后,先查看单选框的状态,如果“选中”属性为真,就允许切换到其他子夹。如果“选中”属性为假,就不允许改变子夹,即不允许切换,如图 8-28 所示。

子程序名	返回值类型	公开	备注
_选择夹1_子夹被改变			

子程序名	返回值类型	公开	备注
_选择夹1_将改变子夹	逻辑型		

如果真 (单选框1. 选中 = 假)

返回 (假)

如果没有选中“是的,非常喜欢,我正在进行”,就不允许切换到其它子夹”

返回“假”就表示不允许改变子夹 (不允许切换)

从这里我们应该得到启示:可以在本事件产生时,检查用户的输入是否满足要求。

图 8-28 “选择夹_将改变子夹”代码示例

【运行结果】运行例程 8-16,观测单击单选框前后标签的变化,如图 8-29 所示。

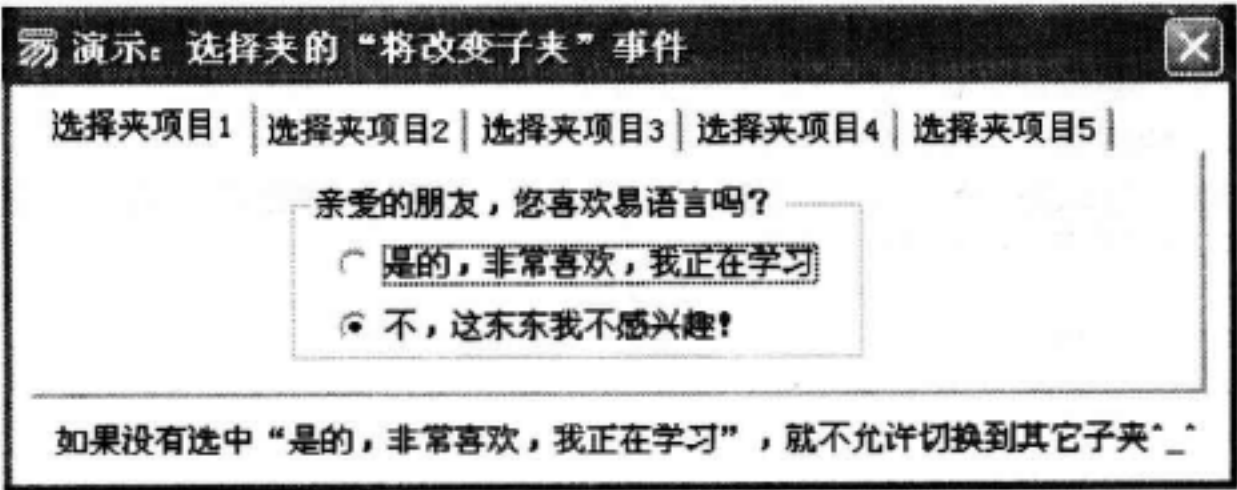





图 8-29 “选择夹_将改变子夹”运行结果示例

【注意】选择夹、选择框都是易语言组件，一字之差，外形却大不相同，功能也有差别。

8.5 分隔条组件

8.5.1 分隔条概述

分隔条组件，可以方便地在程序中实现动态改变组件的位置及大小。程序运行后，根据对分隔条组件“方向”属性的设置，鼠标移动到分隔条上时鼠标指针会改变成“”或“”。当按下鼠标左键并移动时，会有和分隔条组件相同大小的虚线跟随鼠标移动，当鼠标左键被放开则触发分隔条组件的“被拖动”事件，即可在该事件子程序中改变组件的位置及大小。

分隔条没有自己的方法，下面重点介绍分隔条的属性和事件。

8.5.2 分隔条的属性

分隔条的重要属性有“背景颜色”和“方向”等。“背景颜色”属性可以设置分隔条组件的背景颜色。程序中，可以配合程序整体界面的风格和颜色来设置分隔条的颜色。“方向”属性控制分隔条的移动方向，可以设置为“横向”和“纵向”，当设置为“横向”则鼠标拖动分隔条时，分隔条横向水平移动；当设置为“纵向”则分隔条纵向垂直移动。

8.5.3 分隔条的事件

分隔条组件的事件有“被拖动”。当鼠标拖动分隔条移动到新位置后，放开鼠标时触发该事件。该事件有 2 个参数：“原位置”和“目的位置”。“原位置”为分隔条组件移动前的位置，“目的位置”为分隔条组件准备移动到的位置。下面通过一个简单的例子认识其功能。

【范例 8-5】使用分隔条组件的“被拖到”事件来实现学生信息的管理。具体参考例程 8-17。

下面用一个易语言程序来演示使用分隔条组件实现改变组件大小，首先新建一个易程序，在窗口中添加一个树型框组件和一个超级列表框组件，将两个组件放在相邻的位置上，可以在树型框和超级列表框中随意添加一些项目。然后添加一个分隔条组件，将分隔条放在树型框和超级列表框之间，并将“方向”属性设置为“纵向”。如图 8-30 所示。

窗口程序集名	保留	保留	备注
窗口程序集1			

子程序名	返回值类型	公开	备 注		
_分隔条1_被拖动					
参数名	类 型	参考	可空	数组	备 注
原位置	整数型				
目的位置	整数型				

```
树型框1.宽度 = 目的位置 - 树型框1.左边
超级列表框1.移动(目的位置 + 分隔条1.宽度, _启动窗口.宽度 - 树型框1.左边 - 树型框1.宽度, )
    ※左边: 目的位置 + 分隔条1.宽度
    ※顶边:
    ※宽度: _启动窗口.宽度 - 树型框1.左边 - 树型框1.宽度
    ※高度:
分隔条1.左边 = 目的位置
```

图 8-30 “分隔条_被拖动”代码示例

【运行结果】运行例程 8-17, 由于分隔条移动后的位置, 正是树型框右边的位置, 即树型框的宽度和树型框左边的和, 因此, 树型框移动后的宽度等于“目的位置”(即树型框右边的位置)减去树型框的左边位置。

分隔条移动后的位置, 将决定超级列表框左边的位置, 所以分隔条移动后将超级列表框的左边设置为“目的位置”加上分隔条的宽度。分隔条移动后还要改变超级列表框的宽度, 超级列表框的宽度等于启动窗口的宽度减去树型框右边的位置, 即减去树型框左边与宽度之和的位置。运行后的效果如图 8-31 所示。

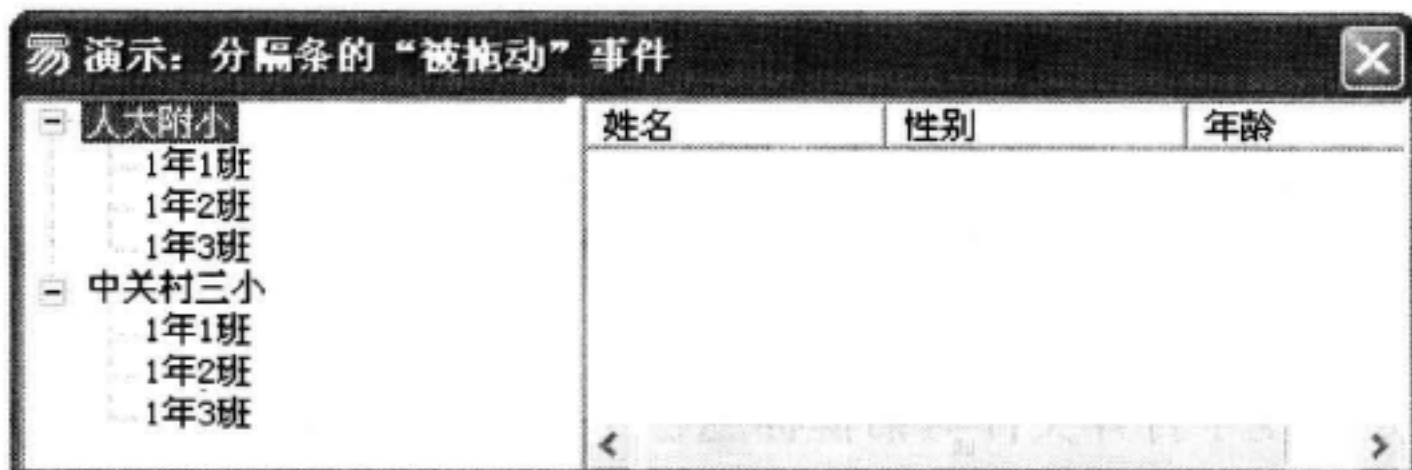



图 8-31 “分隔条_被拖动”运行结果示例

【注意】“目的位置”并不是分隔条已经移动后的位置, 要使分隔条移动, 还要在事件子程序下编写代码。当分隔条组件的“方向”属性为“横向”时, “原位置”和“目的位置”参数为分隔条的顶边值; 当“方向”属性为“纵向”时, “原位置”和“目的位置”为分隔条的左边值。

8.6 通用对话框组件

8.6.1 通用对话框概述

通用对话框  是非可视组件。通过设置它的“类型”属性, 可以被用作“打开文件对话框”、“保存文件对话框”、“字体选择对话框”、“打开帮助对话框”等。在易语言中提供的对话框, 是 Windows 的标准对话框。

通用对话框没有自己的事件, 下面重点介绍通用对话框的属性和方法。

8.6.2 通用对话框的属性

通用对话框的主要属性有“类型”、“文件名”、“过滤器”、“标题”、“默认文件后缀”、“初始目录”、“创建时提示”、“文件必须存在”、“目录必须存在”、“不改变目录”、“文件覆盖提示”、“帮助文件名”、“帮助命令”和“帮助标志值”等。此外还有一些其他属性可以加强通用对话框的技巧应用, 如“字体名称”、“字体大小”、“字体颜色”、“加粗”、“倾斜”、“下划线”、“删除线”等。

通用对话框组件的“类型”属性是最重要的属性, 负责控制打开的对话框的类型。内容为整数型, 有 4 种可选值: “0. 打开文件”、“1. 保存文件”、“2. 字体选择”和“3. 打开帮助”, 默认为“1. 打开文件”。如果需要的是不是“打开文件对话框”(默认类型), 就必须事先设置该属性。当改变不同的“类型”属性时, 会有其他一些属性变灰不可操作, 这是由“类型”属性决定的。如当“类型”属性为“打开文件”时, 下面的“标题”、“文件名”、“过滤器”等属性都有效了。而当“类型”属性为“字体选择”时, 下面的很多属性都无效了。当“类型”属性为“打开帮助”

时,帮助文件名属性有效果,这时要填入一个帮助文件名,否则在运行时无法打开帮助文件。
如图 8-32 所示,具体参考例程 8-18。

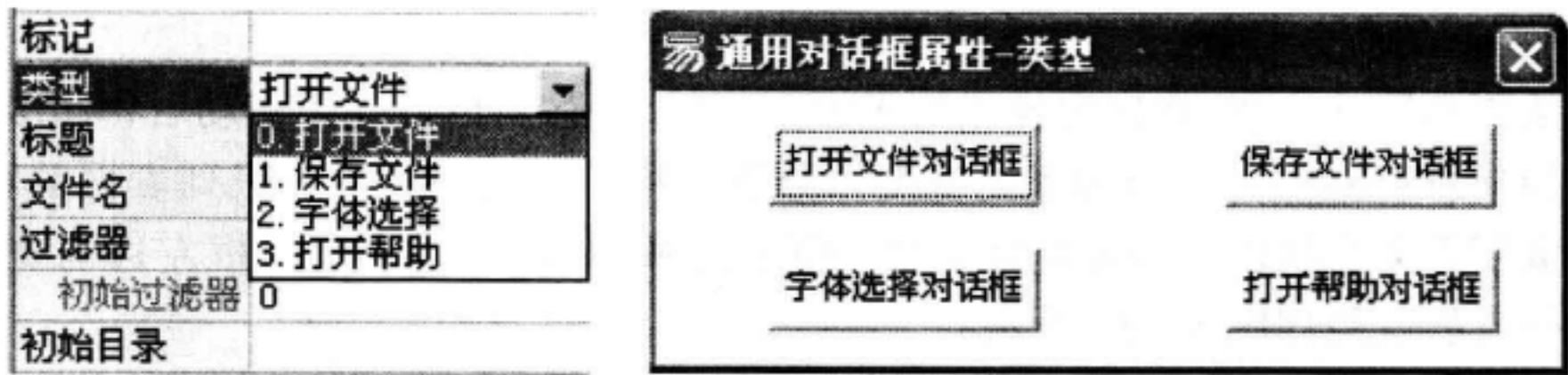


图 8-32 通用对话框“类型”示例

以下根据“类型”属性,依次对各个重要属性进行讲解。

(1) 以下属性在类型 = 0,即当“类型”属性为“打开文件”时,下面的“标题”、“文件名”、“过滤器”等属性都有效了。

“标题”属性是出现于打开文件对话框标题栏中的文字(默认为“打开”)。程序控制代码如下:

通用对话框 1. 标题 = “请先打开一个文本文件”

“文件名”属性用于取得用户在打开文件对话框中指定的文件(含全路径及扩展名)。文件名往往是通过通用对话框取得的,如:

文件名 = 通用对话框 1. 文件名

“过滤器”属性内容为文本型,由单个或多个“成对的文本串”组成。每对文本串由两部分组成,第 1 部分描述显示形式,第 2 部分指定实际的过滤匹配符。各文本串之间用“|”隔开。

【范例 8-6】用通用对话框设计实现打开指定后缀名文件功能。具体参考例程 8-19。

三个通用对话框通过过滤器属性设置打开文件的默认属性。具体的程序代码如图 8-33 所示。

窗口程序集名	保留	保留	备注
窗口程序集1			

子程序名	返回值类型	公开	备注
_按钮1_被单击			

| 通用对话框1.过滤器 = “文本文件(*.txt)|*.txt” | | | |
| 通用对话框1.打开() | | | |

子程序名	返回值类型	公开	备注
_按钮2_被单击			

| 通用对话框2.过滤器 = “文本文件(*.txt)|*.txt|所有文件(*.*)|*.*” | | | |
| 通用对话框2.打开() | | | |

子程序名	返回值类型	公开	备注
_按钮3_被单击			

| 通用对话框3.过滤器 = “网页文件(htm/html)|*.htm;*.html|所有文件(*.*)|*.*” | | | |
| 通用对话框3.打开() | | | |

图 8-33 通用对话框“过滤器”代码示例

【运行结果】运行例程 8-19,观测单击“保存编辑框内容到文件(二)”按钮的变化,如图 8-34所示。



图 8-34 通用对话框“过滤器”运行结果示例

【注意】第3个应用实例中，“*.htm”与“*.html”之间的是半角分号“;”。

(2) 当通用对话框的类型 = 1(保存文件对话框)时,下面的“默认文件后缀”、“初始目录”、“创建时提示”、“文件必须存在”、“目录必须存在”、“不改变目录”和“文件覆盖提示”等属性都有效了。

“默认文件后缀”属性负责指定当用户没有输入文件的后缀名称时自动使用的文件后缀名称。

【范例 8-7】用通用对话框设计实现保存文件的默认文件后缀功能。具体参考例程 8-20。

这里仅提供一个在“保存文件对话框”中使用“默认文件后缀”的例程,其中两个通用对话框控件都已设置成“保存文件”类型(类型 = 1)。通用对话框 2 的“默认文件后缀”属性已设置为“txt”,通用对话框 1 的“默认文件后缀”属性没有设置。单击左边的按钮,将弹出通用对话框 1,随便输入一个不带扩展名的文件名如“保存文件 1”,则文件被存盘为“保存文件 1”;单击右边的按钮,将弹出通用对话框 2,随便输入一个不带扩展名的文件名如“保存文件 2”,则文件被存盘为“保存文件 2.txt”(自动加上了扩展名“txt”)。具体的程序代码如图 8-35 所示。

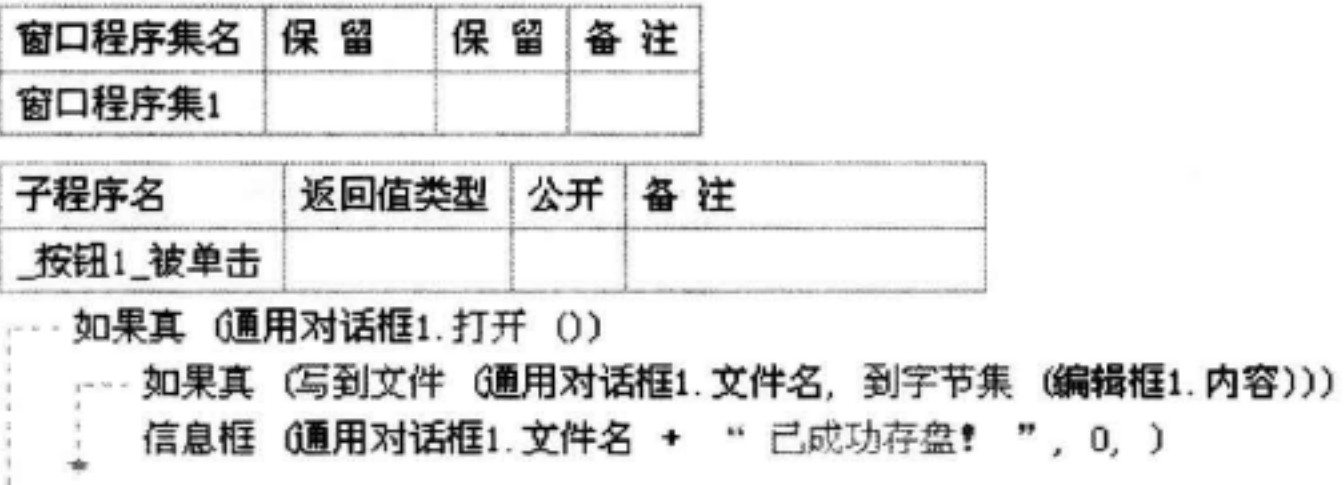


图 8-35 通用对话框“默认文件后缀”代码示例

【运行结果】运行例程 8-20,观测单击“保存编辑框内容到文件(二)”按钮的变化,如图 8-36所示。

【注意】“文件名”、“过滤器”、“初始过滤器”、初始目录、默认文件后缀、目录必须存在、不改变目录”等属性与“打开文件对话框”同名属性含义相同,在类型 = 1,即为“保存文件对话框”时有效。

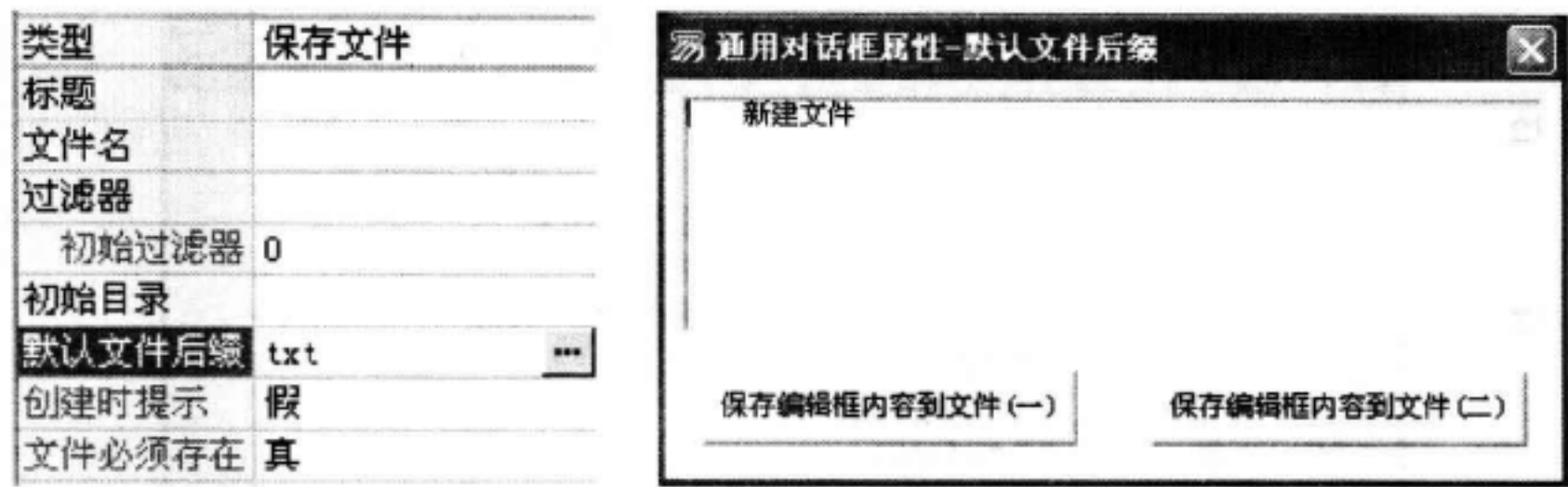


图 8-36 通用对话框“默认文件后缀”运行结果示例

“初始目录”属性即对话框第一次打开时,自动跳转到的目录。可通过在打开对话框前加上一行代码——通用对话框 1. 初始目录 = 取当前目录(),使对话框打开时自动处于当前目录。

【范例 8-8】用通用对话框设计实现打开文件的默认初始目录功能。具体参考例程 8-21。

以下示例窗口中存在 2 个组件:按钮组件和通用对话框组件,点击“使通用对话框打开时,自动切换到系统当前目录”按钮,弹出“打开”框,即可按照用户的需求打开文件。具体的程序代码如图 8-37 所示。

窗口程序集名	保留	保留	备注
窗口程序集1			

子程序名	返回值类型	公开	备注
_按钮1_被单击			

通用对话框1.初始目录 = 取当前目录 0
通用对话框1.打开 0

图 8-37 通用对话框“初始目录”代码示例

【运行结果】运行例程 8-21,观测单击“使通用对话框打开时,自动切换到系统当前目录”按钮的变化,如图 8-38 所示。

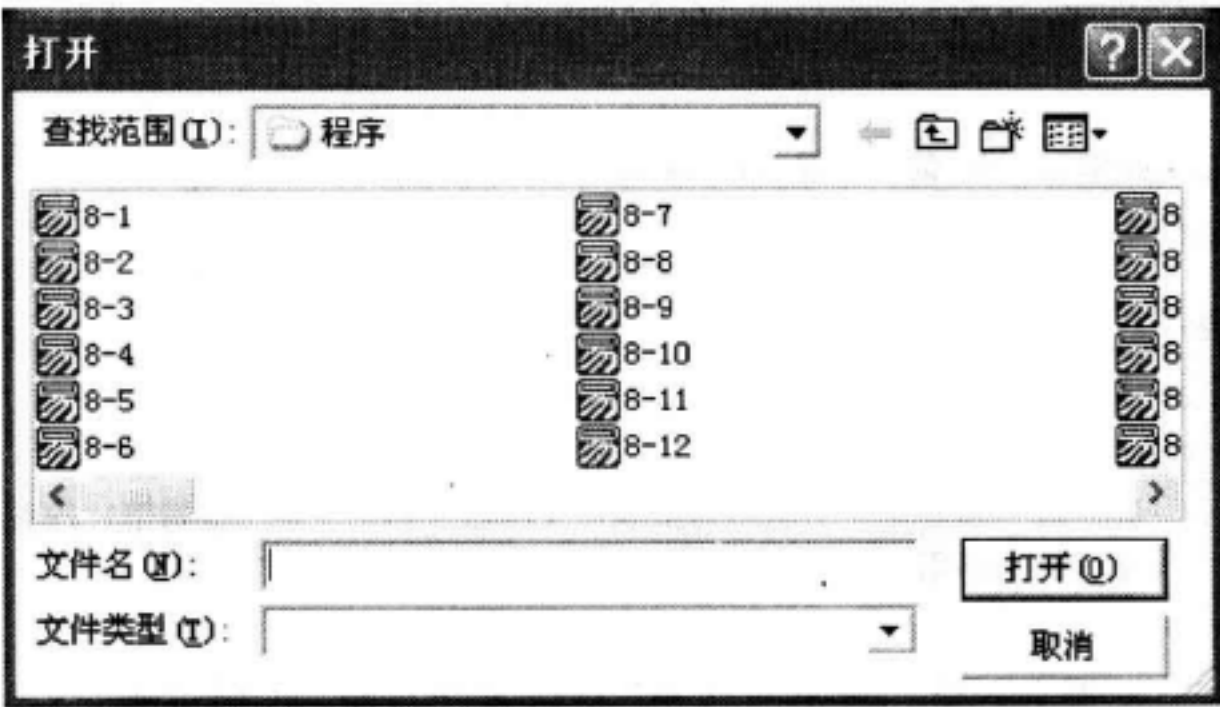


图 8-38 通用对话框“初始目录”运行结果示例

【注意】通过单击按钮,可以自动切换到系统当前目录。即在当前目录中打开文件,而不是系统默认的“我的文档”文件夹。

“创建时提示”属性是如果用户指定了一个不存在的文件名称,负责是否提示用户创建它。内容为逻辑型,只能为“真”或“假”,默认为“假”。默认情况下,如果用户指定了不存在

的文件名,只会警告说“找不到”指定文件。设置该属性为“真”后,如果用户指定了不存在的文件名,除了警告“找不到”指定文件外,还将询问是否创建新文件。

【范例 8-9】用通用对话框设计实现创建时提示功能。具体参考例程 8-22。

在打开一个文件时,直接输入一个当前目录下没有的文件名,单击“确定”后就会弹出一个对话框,提示用户是否创建该文件。具体的程序代码如图 8-39 所示。

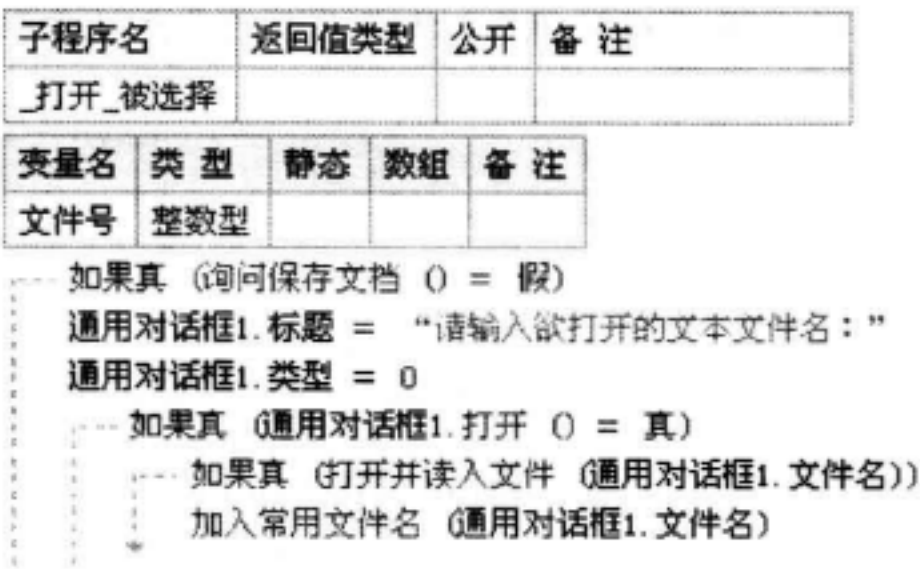


图 8-39 通用对话框“创建时提示”代码示例

【运行结果】运行例程 8-22,观测单击“打开”按钮的变化,如图 8-40 所示。

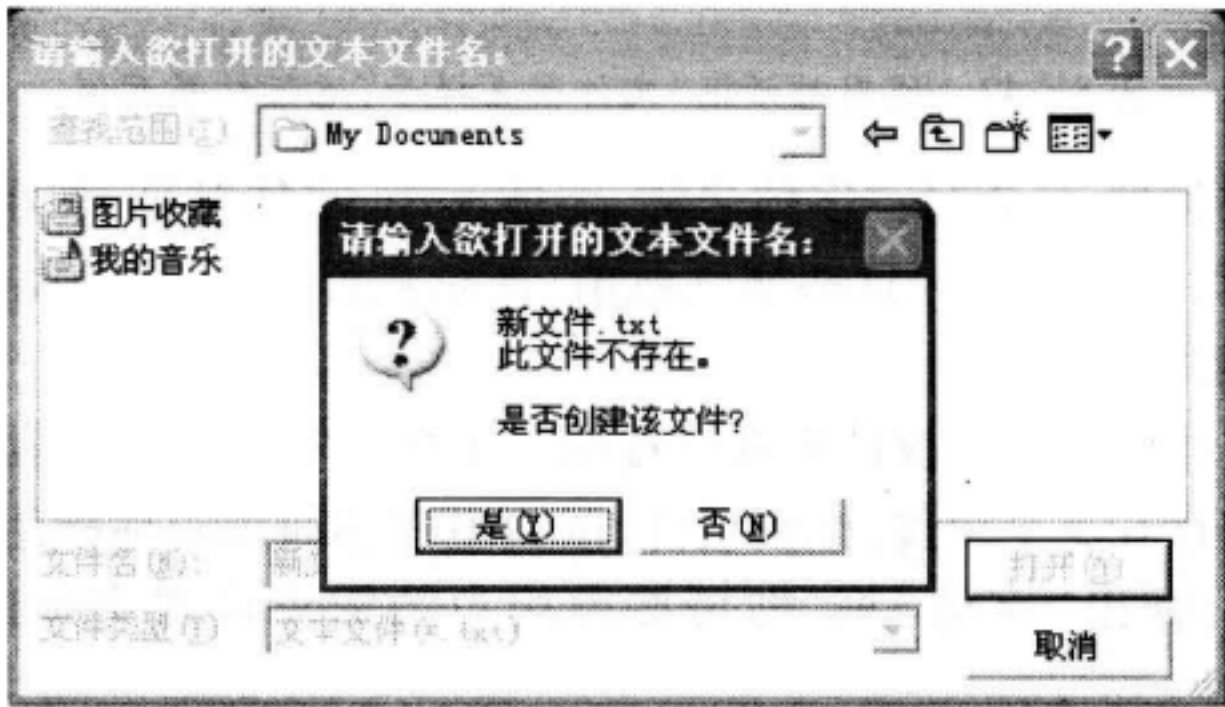


图 8-40 通用对话框“创建时提示”运行结果示例

【注意】以上“创建时提示”属性在类型 = 1,即为“保存文件对话框”时有效。

“文件必须存在”、“目录必须存在”属性表示是否允许用户指定一个不存在的文件或目录。内容为逻辑型,只能为“真”或“假”,默认值均为“真”。

“不改变目录”属性在对话框关闭后再次打开时,是否自动返回到“第一次打开对话框时的”文件目录。内容为逻辑型,只能为“真”或“假”,默认为“假”。当打开文件时,选择了与运行时不同的目录下的其他文件,那么它就会将当前目录定位在新的目录下,如果使本项属性为“真”,那么无论文件在任何目录中,当前目录也不改变。

“文件覆盖提示”属性指当用户指定了一个已经存在的文件,询问用户是否覆盖此文件。内容为逻辑型,只能为“真”或“假”,默认值为“真”。

【范例 8-10】用通用对话框设计实现文件覆盖提示功能。具体参考例程 8-23。

在另存一个文件时,如该文件已经存在时,会弹出一个是否要替换的提示。具体的程序代码如图 8-41 所示。

【运行结果】运行例程 8-23,观测单击“另存为”按钮的变化,如图 8-42 所示。

【注意】以上“文件覆盖提示”属性在类型 = 1,即为“保存文件对话框”时有效。

子程序名	返回值类型	公开	备注
另存为_被选择			

```
通用对话框1.标题 = "请输入欲另存到的文本文件名："
通用对话框1.类型 = 1
如果真 (通用对话框1.打开 () = 真)
    保存文档 (通用对话框1.文件名)
```

图 8-41 通用对话框“文件覆盖提示”代码示例

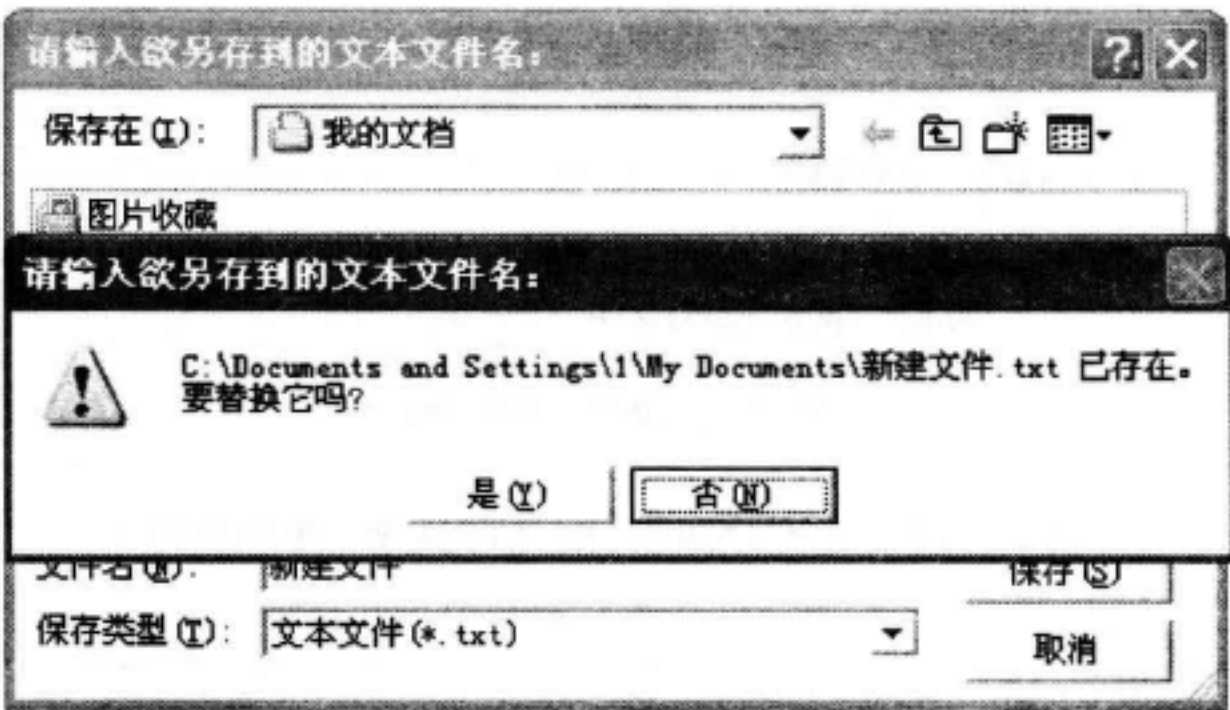


图 8-42 通用对话框“文件覆盖提示”运行结果示例

(3) 以下属性在类型 = 2 时,“字体名称、字体大小、字体颜色、加粗、倾斜、下划线、删除线”属性有效,使用时,通常不是为这些属性赋值,而是在程序使用者单击“字体选择对话框”中的“确定”按钮后读取它们的值。

【范例 8-11】用通用对话框设计实现设置指定字体的外观。具体参考例程 8-24。

以下示例窗口中存在 3 个组件:标签组件、按钮组件和通用对话框组件。点击“设置标签中文字的字体”按钮,弹出“字体”框,即可按照用户的需求设置字体的格式。如图 8-43 所示。

窗口程序集名	保留	保留	备注
窗口程序集1			

子程序名	返回值类型	公开	备注
_按钮1_被单击			

```
如果真 (通用对话框1.打开 ())
    标签1.字体.字体名称 = 通用对话框1.字体名称
    标签1.字体.字体大小 = 通用对话框1.字体大小
    标签1.字体.加粗 = 通用对话框1.加粗
    标签1.字体.倾斜 = 通用对话框1.倾斜
    标签1.字体.下划线 = 通用对话框1.下划线
    标签1.字体.删除线 = 通用对话框1.删除线

    标签1.文本颜色 = 通用对话框1.字体颜色 不存在“标签1.字体.字体颜色”之说。标签文字的颜色由其“文本颜色”属性控制。
```

图 8-43 通用对话框字体设置代码示例

【运行结果】运行例程 8-24,观测单击“设置标签中文字的字体”按钮的变化,如图 8-44 所示。

【注意】以上“字体名称、字体大小、字体颜色、加粗、倾斜、下划线、删除线”属性在类型 = 2,即为“字体选择对话框”时有效。

(4) 以下属性在类型 = 3,即为“打开帮助对话框”时有效。“帮助文件名”和“帮助命令”

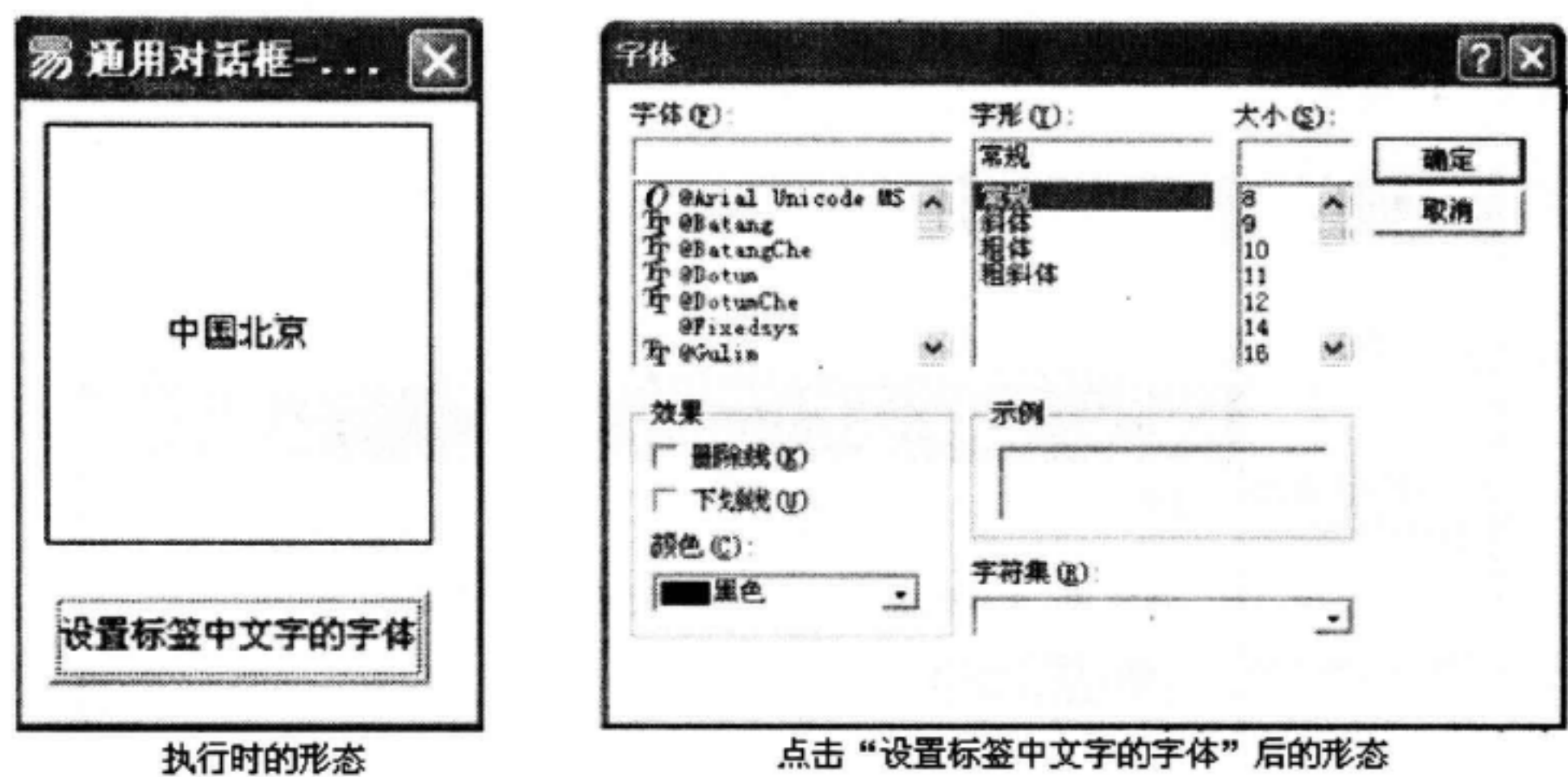


图 8-44 通用对话框字体设置运行结果示例

属性,即在此可以填入一个帮助文件名,此帮助文件必须是以“. hlp”为类型的帮助文件。
【范例 8-12】用通用对话框实现帮助文件的相关操作。具体参考例程 8-25。
以下示例窗口中存在 5 个组件,1 个通用对话框组件和 4 个按钮组件,点击“显示帮助文件使用方法”按钮,弹出“帮助主题:如何使用帮助”框,即可按照用户的需求查看相关帮助。如图 8-45 所示。

窗口程序集名	保留	保留	备注
窗口程序集1			

子程序名	返回值类型	公开	备注
_按钮1_被单击			

通用对话框1.帮助命令 = 0
通用对话框1.打开 ()

子程序名	返回值类型	公开	备注
_按钮2_被单击			

通用对话框1.帮助命令 = 3
通用对话框1.打开 ()

子程序名	返回值类型	公开	备注
_按钮3_被单击			

通用对话框1.帮助命令 = 4
通用对话框1.打开 ()

子程序名	返回值类型	公开	备注
_按钮4_被单击			

通用对话框1.帮助命令 = 5
通用对话框1.打开 ()

图 8-45 通用对话框帮助代码示例

【运行结果】运行例程 8-25,第一个按钮被单击后,会直接弹出一个帮助文件的帮助主题查找方式。第二个按钮被单击后,会弹出帮助文件的索引主题。第三个按钮被单击后,会弹出如何使用帮助文件的一个系统帮助文件,此文件是系统自带的,以说明如何使用帮助文件。第

四个按钮被单击后,会将弹出的帮助文件关闭。观测单击“显示帮助文件使用方法”按钮的变化,如图 8-46 所示。

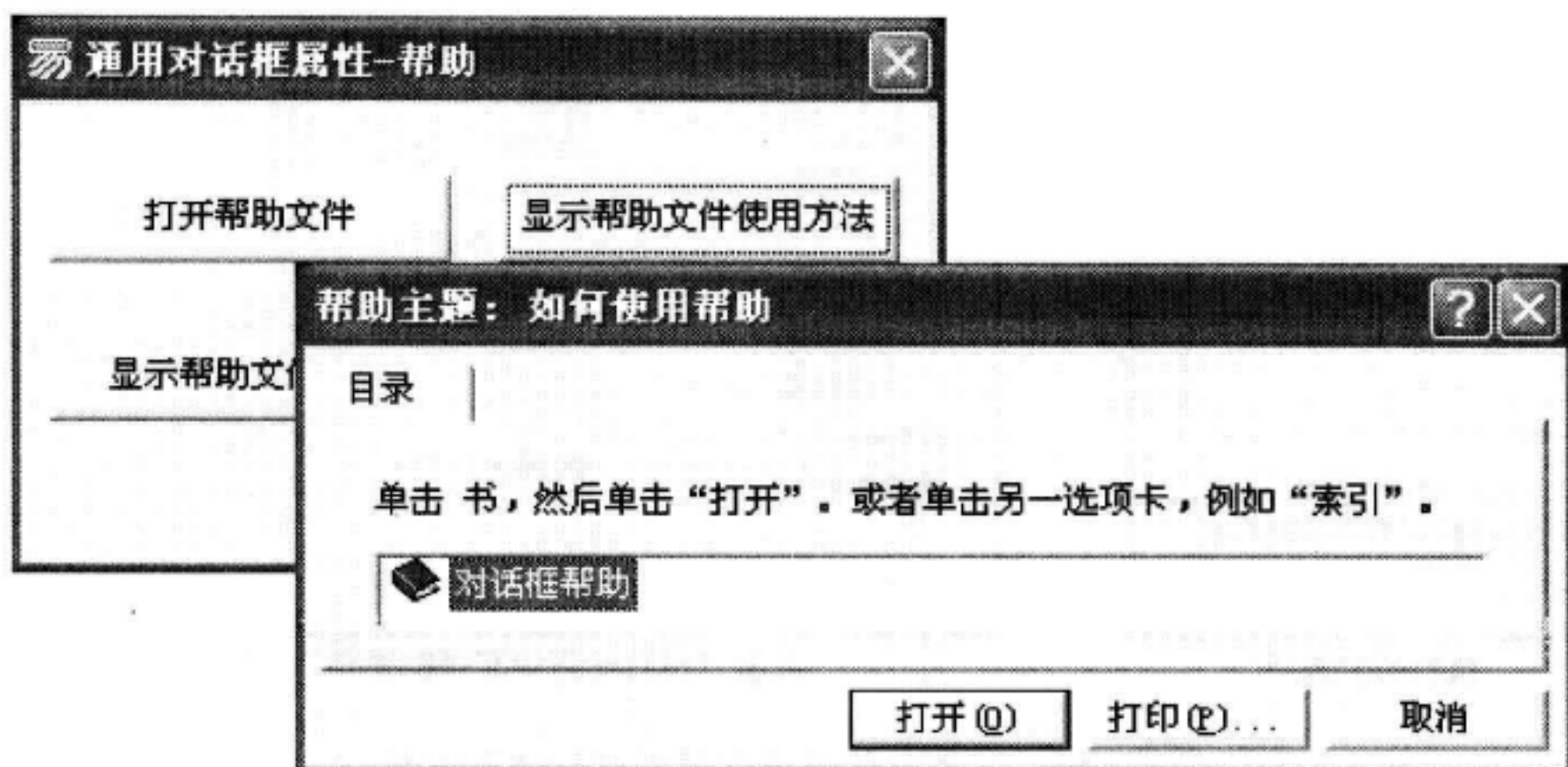


图 8-46 通用对话框帮助运行结果示例

【注意】以上“帮助文件名”和“帮助命令”属性在类型 = 3, 即为“打开帮助对话框”时有效。

“帮助标志值”属性,如果此 HLP 帮助文件中设置了标志值,那么可以通过将帮助命令设置为“1. 显示指定标志信息”,“2. 弹出指定标志信息”,即可以在此属性中输入相应的标志值,从而很快地定位到要帮助的主题中。这对于有针对性地提供帮助是非常好的。

8.6.3 通用对话框的方法

通用对话框的主要方法有“打开()”,负责打开通用对话框(把对话框显示到屏幕上),返回一个逻辑值。对于类型为“打开文件”、“保存文件”、“字体选择”的对话框,如果为“真”,表示用户已通过该对话框输入了有效数据,否则表示用户取消了该对话框,且没有输入任何有效数据。对于类型为“打开帮助”的对话框,如打开帮助成功则返回“真”,否则返回“假”。

“打开()”方法在前面的例程已用到了,这里就不再赘述,具体请参考【范例 8-9】和【范例 8-10】。在此例程中,可以看到如何将打开的历史记录保存下来,以及只使用一个通用对话框即实现了保存与打开的两种功能。

8.7 列表框组件


8.7.1 列表框概述

列表框^①提供一个项目列表,用户可以从选择一个或多个项目。它可以将选择项目全部显示,即在选择项目一次显示不完时,会自动增加滚动条。也可以将常用的一些选择内容呈列出来供挑选。它不带有下拉按钮,虽不节约空间,但最直观。列表框具有数据绑定特性,能够访问大多数标准类型数据库的信息。

下面重点介绍列表框的属性、方法和事件。

8.7.2 列表框的属性

列表框的重要属性有“列表项目”、“现行选中项”、“允许选择多行”、“行间距”、“多列”、“自动排序”、“列表数值”等。

“列表项目”属性用于在界面设计阶段设置列表框的原始项目。可以在列表框上用鼠标右键弹出“设置列表项目属性”菜单,或直接在属性面板的“列表项目”属性中单击按钮,也可以弹出相应的对话框,如图 8-47 所示,具体参考例程 8-26。

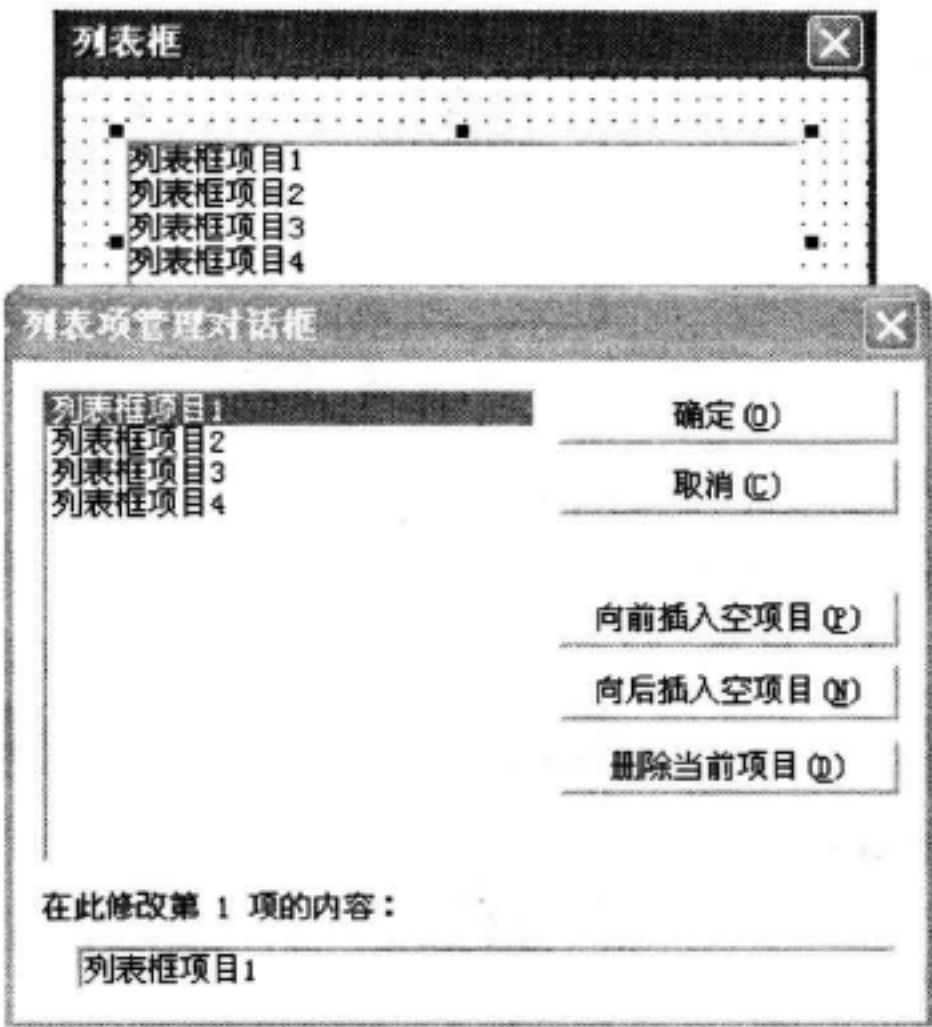


图 8-47 列表项管理对话框

“现行选中项”属性是指定列表框中现行被选中列表项目的位置,位置值从 0 开始,-1 表示现行没有被选中的列表项。如果“允许选择多项”属性为真,则本属性无效,其值恒为 -1。可在设计期设置或在运行期设置和读取该属性的值。图 8-48 表示是当“现行选中项”属性为“假”时的情况,具体参考例程 8-27。

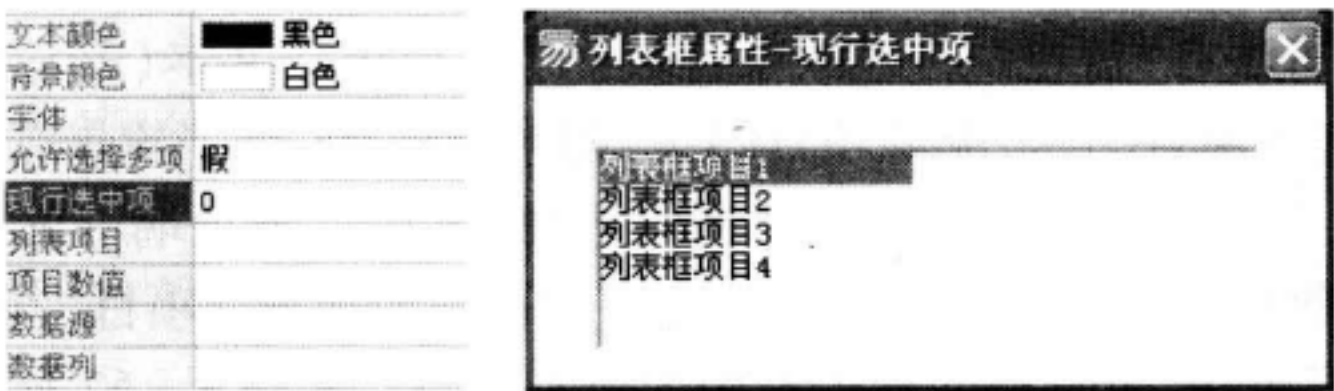


图 8-48 列表框“现行选中项”

“允许选择多行”属性内容为逻辑型,只能为“真”或“假”,默认为“假”。如果本项属性为“真”,那么在程序运行时,可以通过按住 Shift 键或 Ctrl 键后,再用鼠标点选,即可同时选中多项。当本属性为“假”时,表示只能选择一行,这时可使用“取项目文本(现行选中项())”方法取项目内容,取出的是一行文本,当本属性为“真”时,表示可以选择多行项目,但这时要使用“取所有被选择项目()”方法取项目内容,并且取出的是文本数组。图 8-49 表示是当“允许选择多项”属性为“真”时的情况,具体参考例程 8-28。

“多列”属性负责控制列表框是否以多列显示。内容为逻辑型,只能为“真”或“假”,默认为“假”,图 8-50 表示是当“多列”属性为“真”时的情况,具体参考例程 8-29。

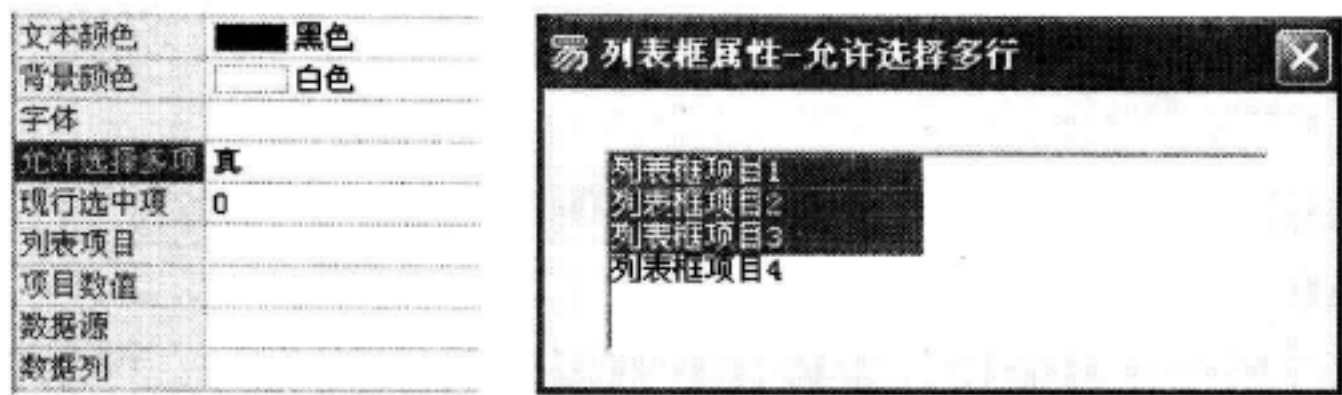


图 8-49 列表框“允许选择多行”

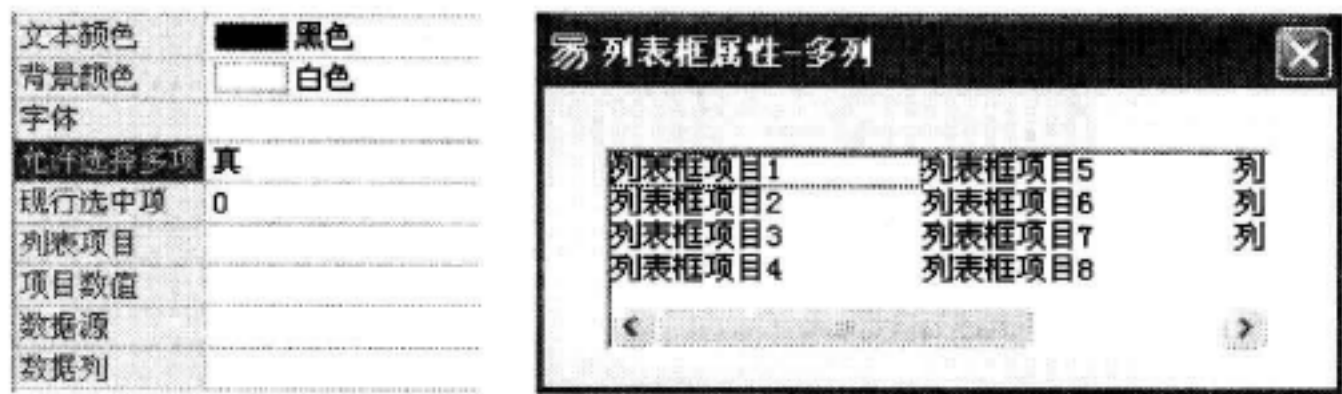


图 8-50 列表框“多列”

“行间距”属性指定列表框的各行之间的间距。内容为整数型,默认值为1。

“自动排序”属性控制是对列表框中的各项目自动排序。内容为逻辑型,只能为“真”或“假”,默认为“假”,图8-51表示是当“自动排序”属性为“真”时的情况,具体参考例程8-30。

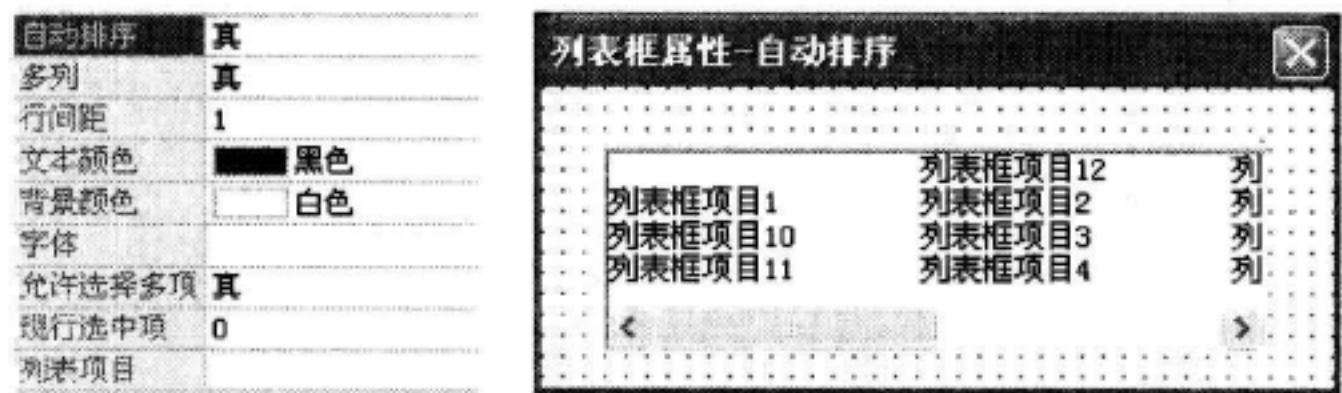


图 8-51 列表框“自动排序”

“列表数值”属性指定与各列表项目相关联的数值,该数值在程序中可以取出或修改。在以上属性中,“列表项目”、“现行选中项”最为重要,初学者要重点掌握。

8.7.3 列表框的方法

列表框的重要方法有“清空()”、“取项目数()”、“取项目文本()”、“置项目文本()”、“加入项目()”、“插入项目()”、“删除项目()”、“取焦点项目()”、“置焦点项目()”、“选择项目()”、“选择()”、“是否被选择()”、“取顶端可见项目()”、“置顶端可见项目()”、“取项目数值()”、“置项目数值()”、“取已选择项目数()”、“取所有被选择项目()”等。

(1) “清空()”方法,负责清除列表框中的所有项目。如图8-52所示,具体参考例程8-31。

列表框1. 清空()

(2) “取项目数()”方法,负责返回当前项目的总数。如图8-53所示,具体参考例程8-32。

编辑框1. 内容 = “当前项目总数为:” + 到文本(列表框1. 取项目数())

(3) “取项目文本()”方法,负责取指定的项目的文本。如图8-54所示,具体参考例程8-33。

编辑框1. 内容 = “取第三个项目的文本为:” + 到文本(列表框1. 取项目文本(2))

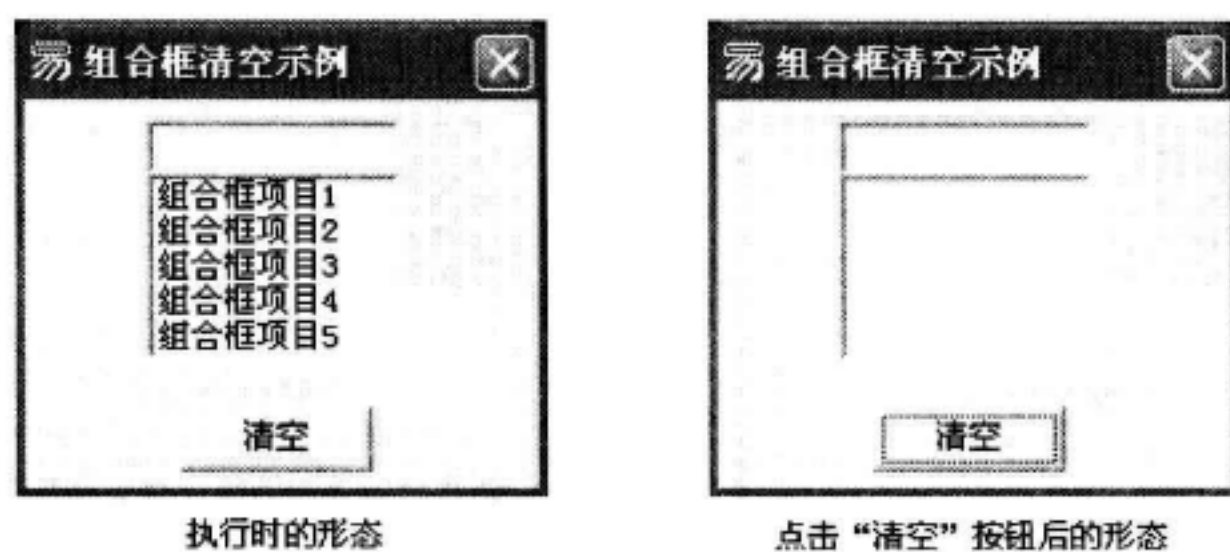


图 8-52 列表框“清空”示例

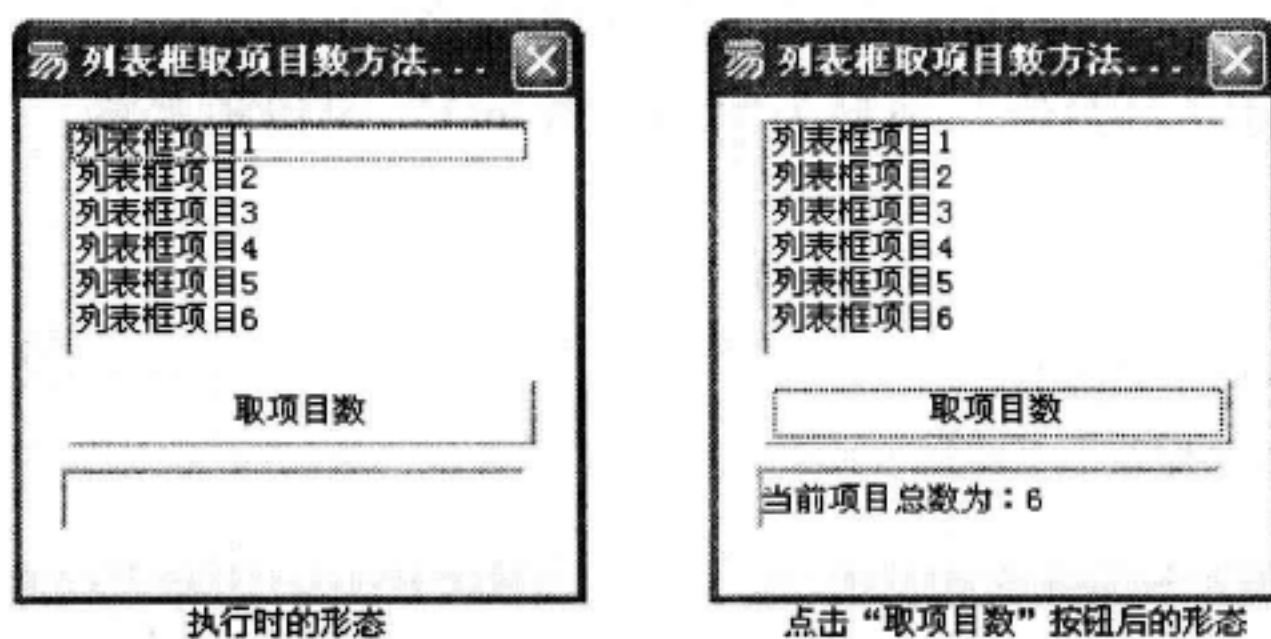


图 8-53 列表框“取项目数”示例

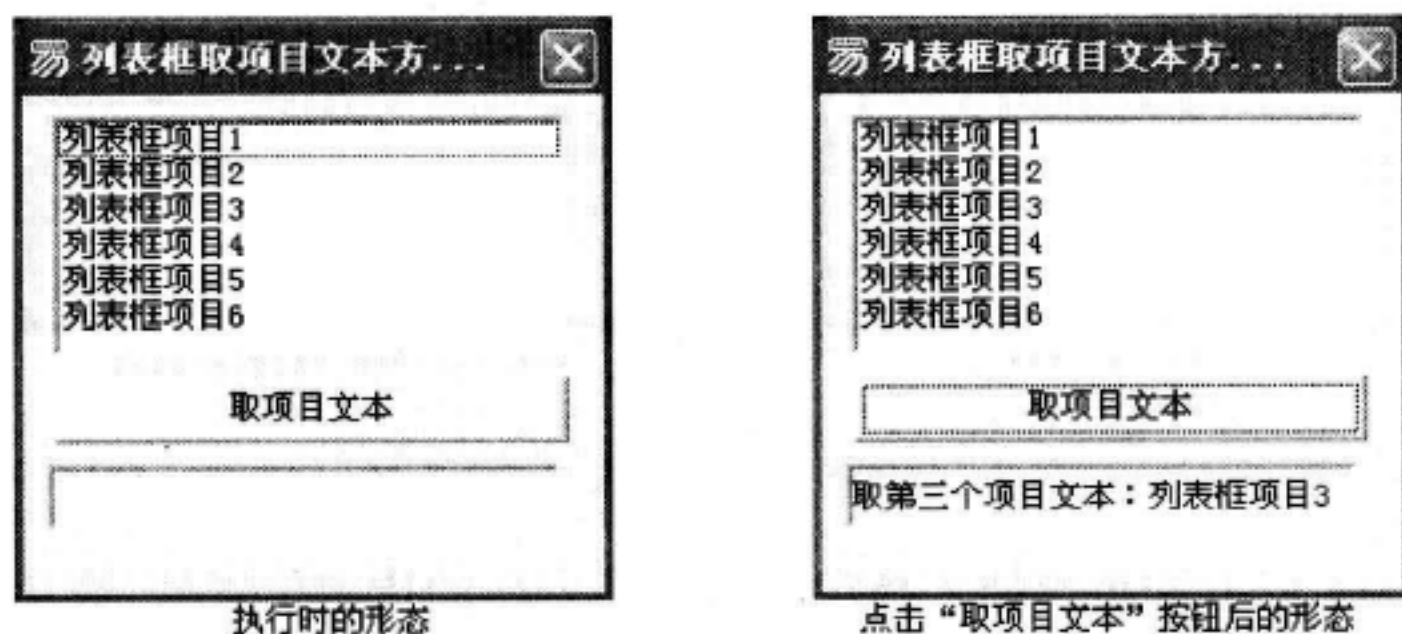


图 8-54 列表框“取项目文本”示例

(4) “置项目文本()”方法,负责设置指定的项目的文本。如图 8-55 所示,具体参考例程 8-34。

列表框 1. 置项目文本(3,“我爱易语言”)

编辑框 1. 内容 = “将项目 4 改文本标题为:我爱易语言”

(5) “加入项目()”方法负责加入指定项目到列表框的尾部,成功返回加入后该项目所处的位置,失败返回 -1。

“加入项目()”调用格式:

〈整数型〉对象. 加入项目(欲加入项目的文本,[与欲加入项目相关的数值])

参数说明:

“欲加入项目的文本”,类型为“文本型(text)”。

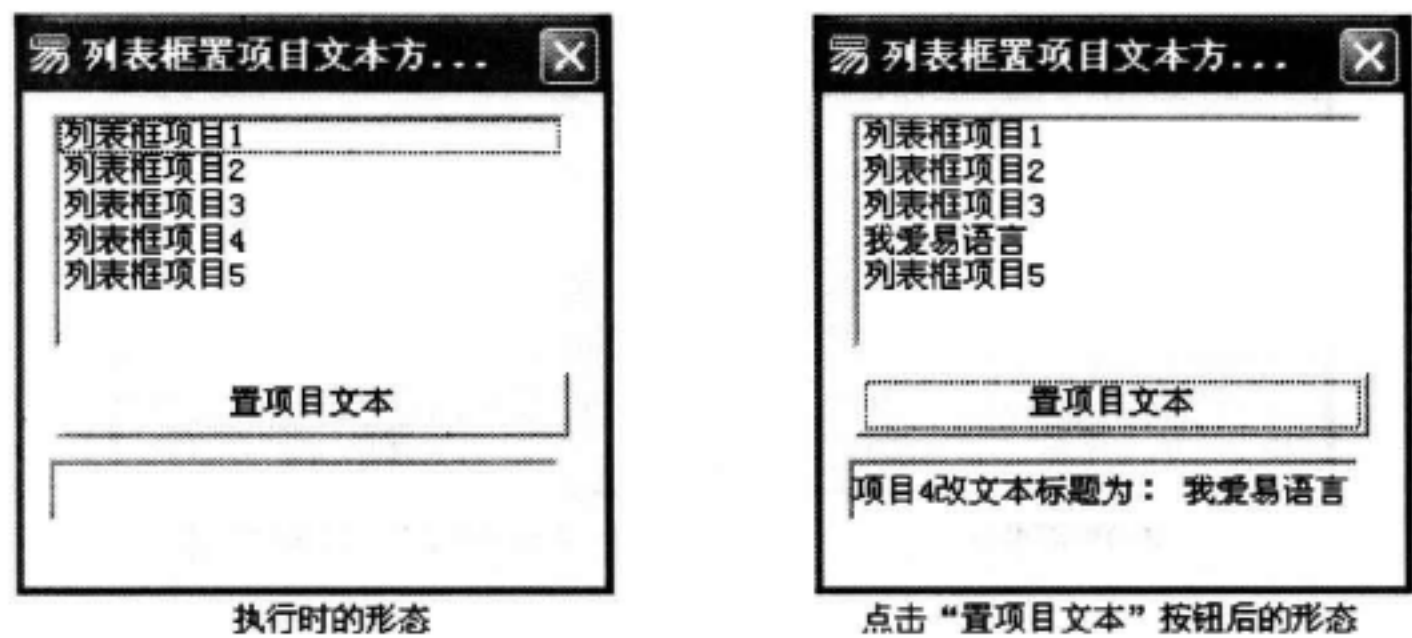


图 8-55 列表框“置项目文本”示例

“与欲加入项目相关的数值”,类型为“整数型(int)”,可以被省略。如果省略本参数,默认值为0。

如图 8-56 所示,具体参考例程 8-35。

列表框 1. 加入项目(“易语言”,)
编辑框 1. 内容 = “加入一个项目”

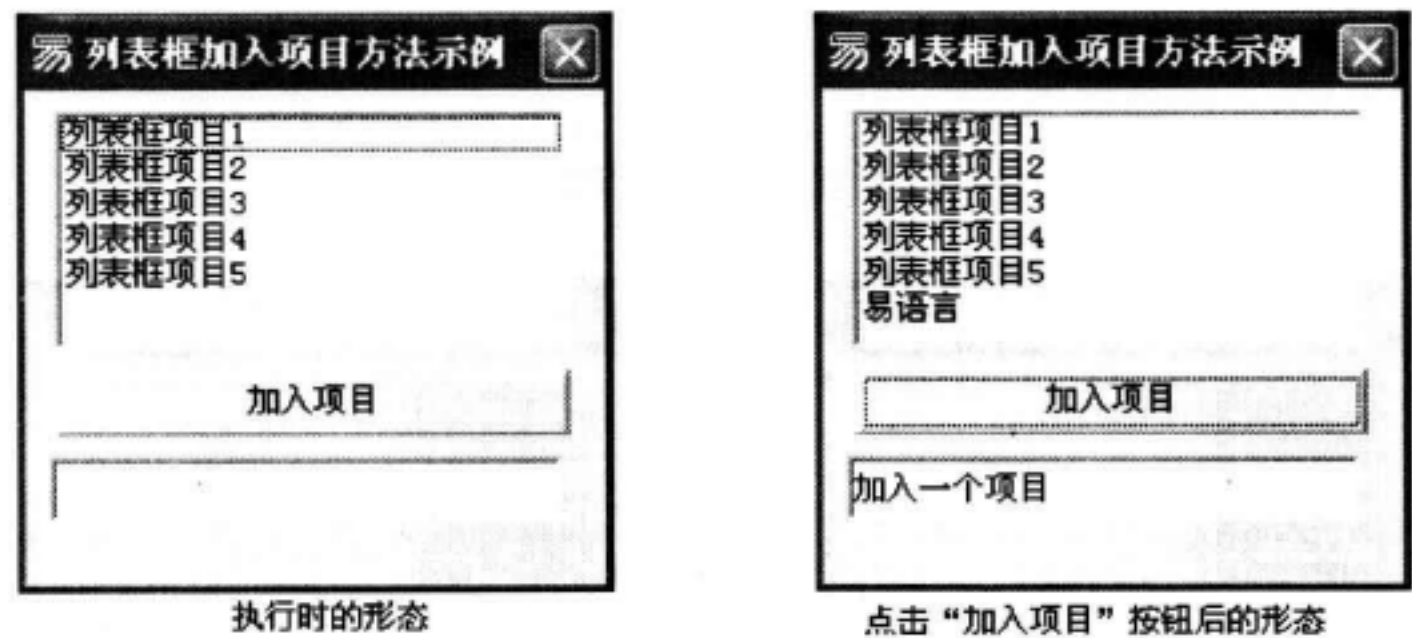


图 8-56 列表框“加入项目”示例

(6) “插入项目()”方法负责插入指定项目到列表框的指定位置处,成功返回插入后该项目所处的位置,失败返回-1。本命令为初级对象成员命令。

“插入项目()”调用格式:

〈整数型〉对象. 插入项目(欲插入的位置, 欲插入项目的文本, [与欲插入项目相关的数值])

参数说明:

“欲插入的位置”,类型为“整数型(int)”。0 为项目位置一,1 为项目位置二,如此类推。

“欲插入项目的文本”,类型为“文本型(text)”。

“与欲插入项目相关的数值”,类型为“整数型(int)”,可以被省略。如果省略本参数,默认值为0。

如图 8-57 所示,具体参考例程 8-36。

列表框 1. 插入项目(2,“我爱易语言”,)
编辑框 1. 内容 = “在第 3 排插入一个项目”

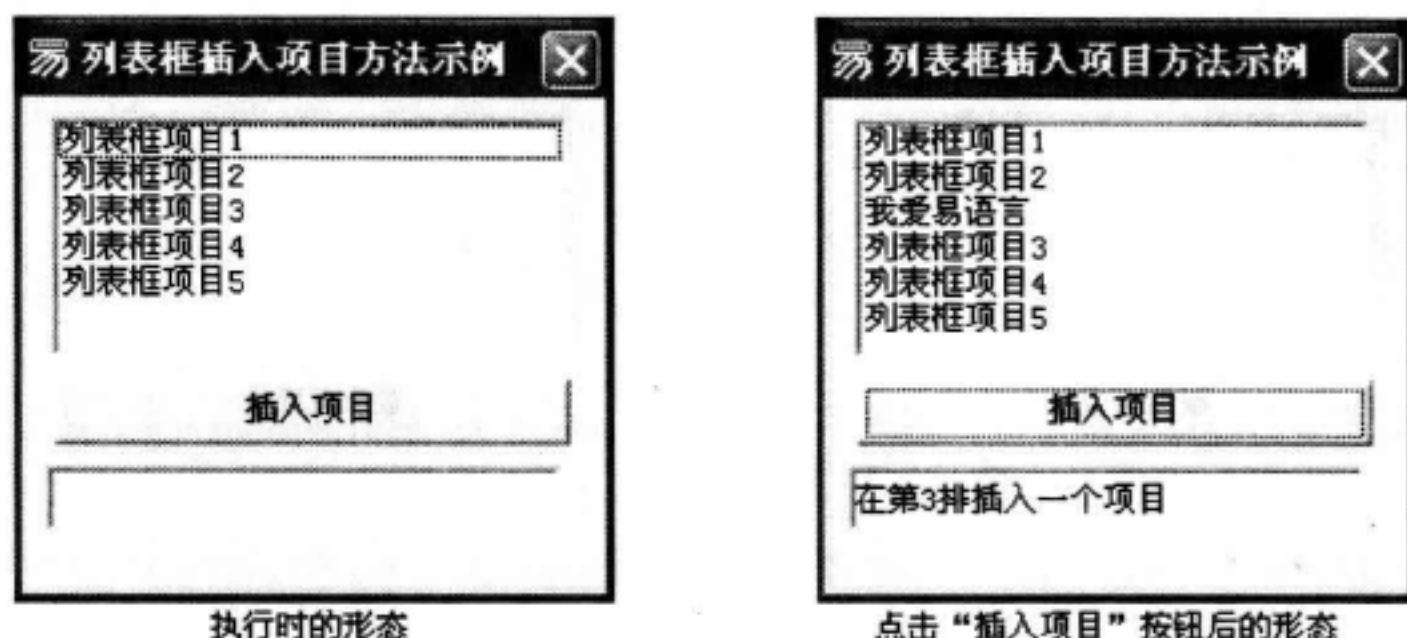


图 8-57 列表框“插入项目”示例

(7) “删除项目()”方法负责删除列表框指定位置处的项目。成功返回“真”,失败返回“假”。本命令为初级对象成员命令。

调用格式:

〈逻辑型〉对象. 删除项目(项目索引)

删除列表框指定位置处的项目。成功返回“真”,失败返回“假”。本命令为初级对象成员命令。

参数说明:

“项目索引”,类型为“整数型(int)”。0 为项目一,1 为项目二,如此类推。

如图 8-58 所示,具体参考例程 8-37。

列表框 1. 删除项目(1)

编辑框 1. 内容 = “删除第 2 个项目”

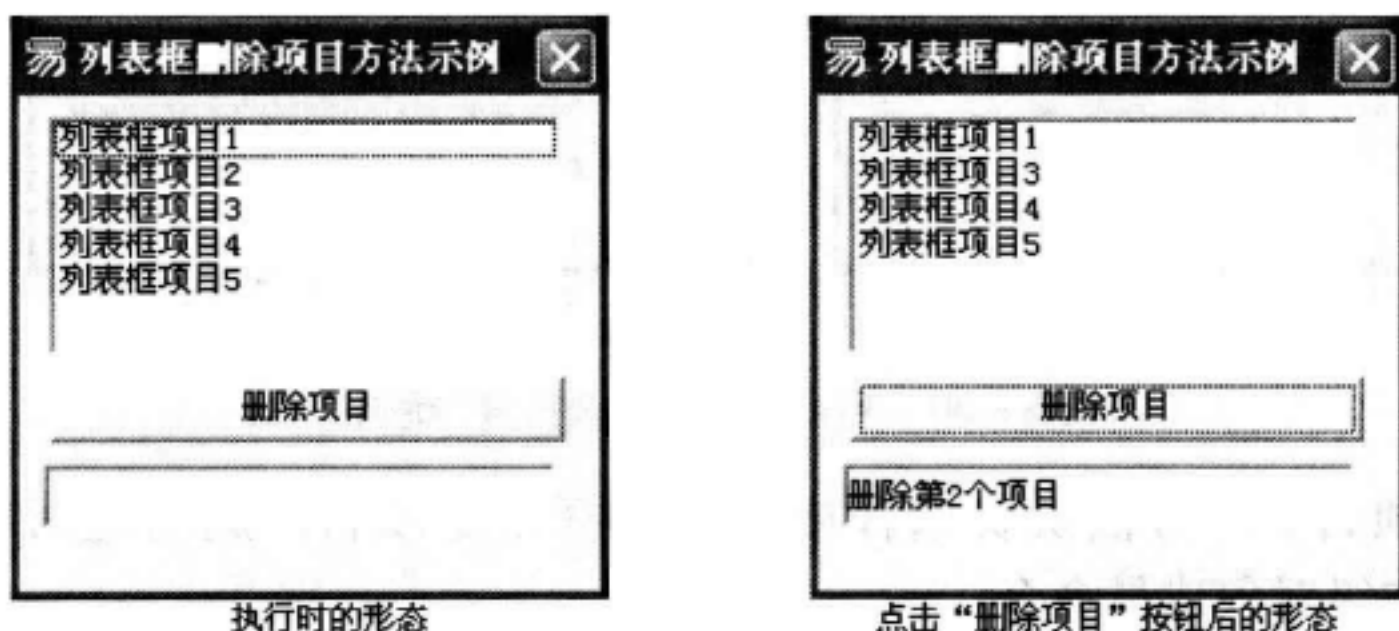


图 8-58 列表框“删除项目”示例

(8) “取焦点项目()”方法仅在多选列表框中使用,负责返回当前焦点项目的位置索引。如果在单选列表框中使用本命令,将返回当前被选择项目的位置索引。

调用格式:

〈整数型〉对象. 取焦点项目()

如图 8-59 所示,具体参考例程 8-38。

编辑框 1. 内容 = 到文本(列表框 1. 取焦点项目())

(9) “置焦点项目()”方法仅在多选列表框中使用,负责设置当前焦点项目。如果在单选列表框中使用本命令,将设置当前被选中项目。成果返回“真”,失败返回“假”。

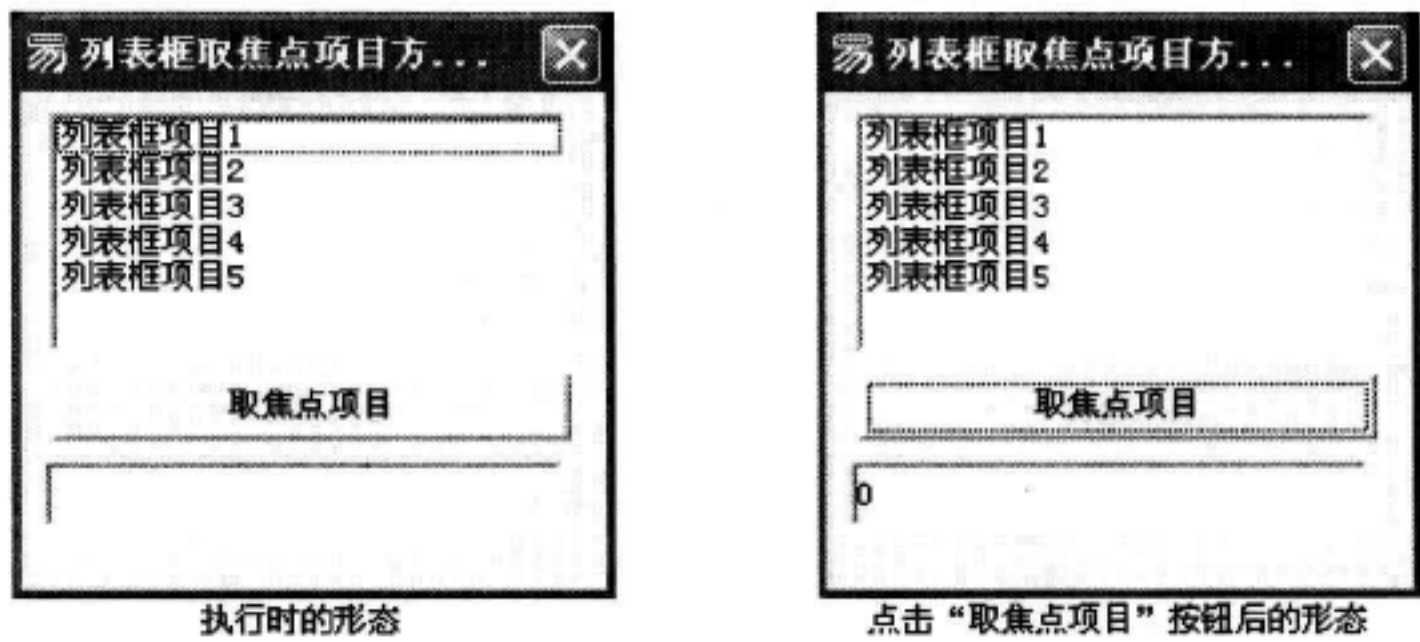


图 8-59 列表框“取焦点项目”示例

调用格式：
〈逻辑型〉对象. 置焦点项目(项目索引)
参数说明：
“项目索引”，类型为“整数型(int)”。0 为项目一，1 为项目二，如此类推。
如图 8-60 所示，具体参考例程 8-39。

```
输入框(“请输入要查询的项目号：”,,, 变量1,)  
编辑框 1. 内容 = 到文本(列表框 1. 置焦点项目(到数值(变量1)))
```

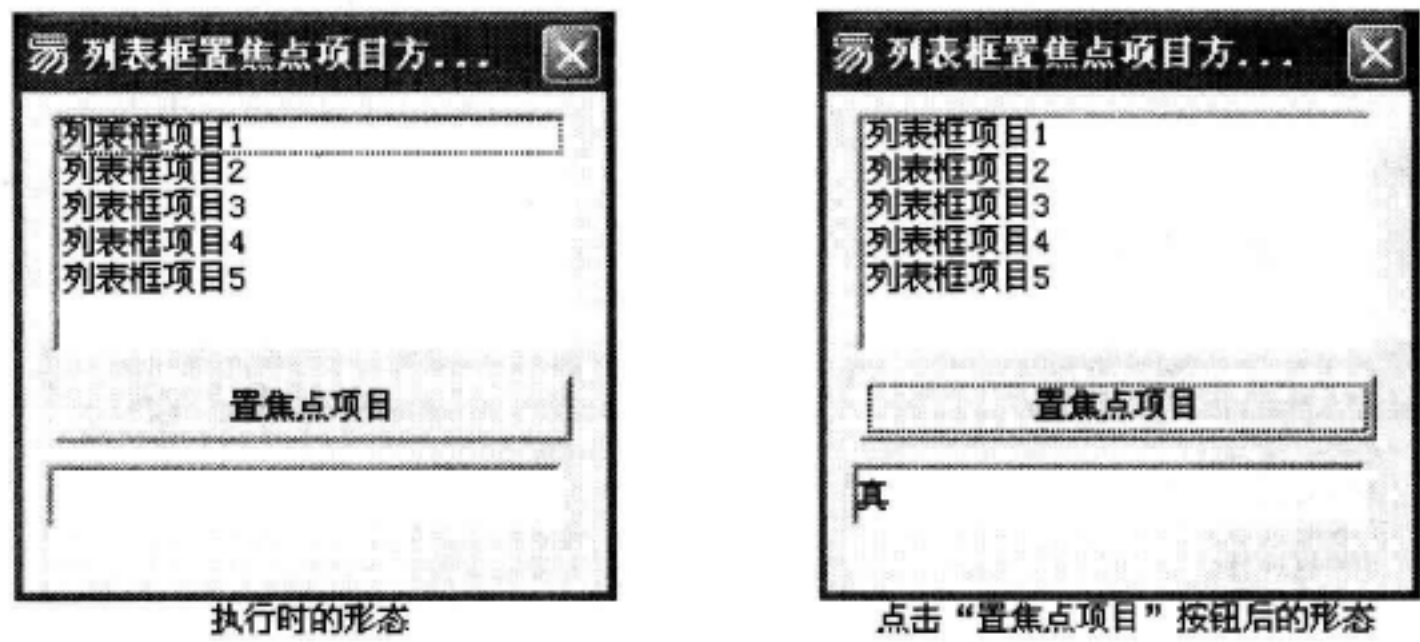


图 8-60 列表框“置焦点项目”示例

(10) “选择项目()”方法负责选择或取消选择指定项目。成功返回“真”，失败返回“假”。本命令为初级对象成员命令。

调用格式：
〈逻辑型〉对象. 选择项目(欲选择或取消选择的项目索引,[欲置入的项目选择状态])
参数说明：
“欲选择或取消选择的项目索引”，类型为“整数型(int)”。0 为项目一，1 为项目二，如此类推。

“欲置入的项目选择状态”，类型为“逻辑型(bool)”，可以被省略。如果省略本参数或者参数值为真，则选择该项目，否则取消对该项目的选择。

如图 8-61 所示，具体参考例程 8-40。

```
输入框(“请输入要选择的的项目号：”,,, 变量1,)  
编辑框 1. 内容 = 到文本(列表框 1. 选择项目(到数值(变量1),))
```

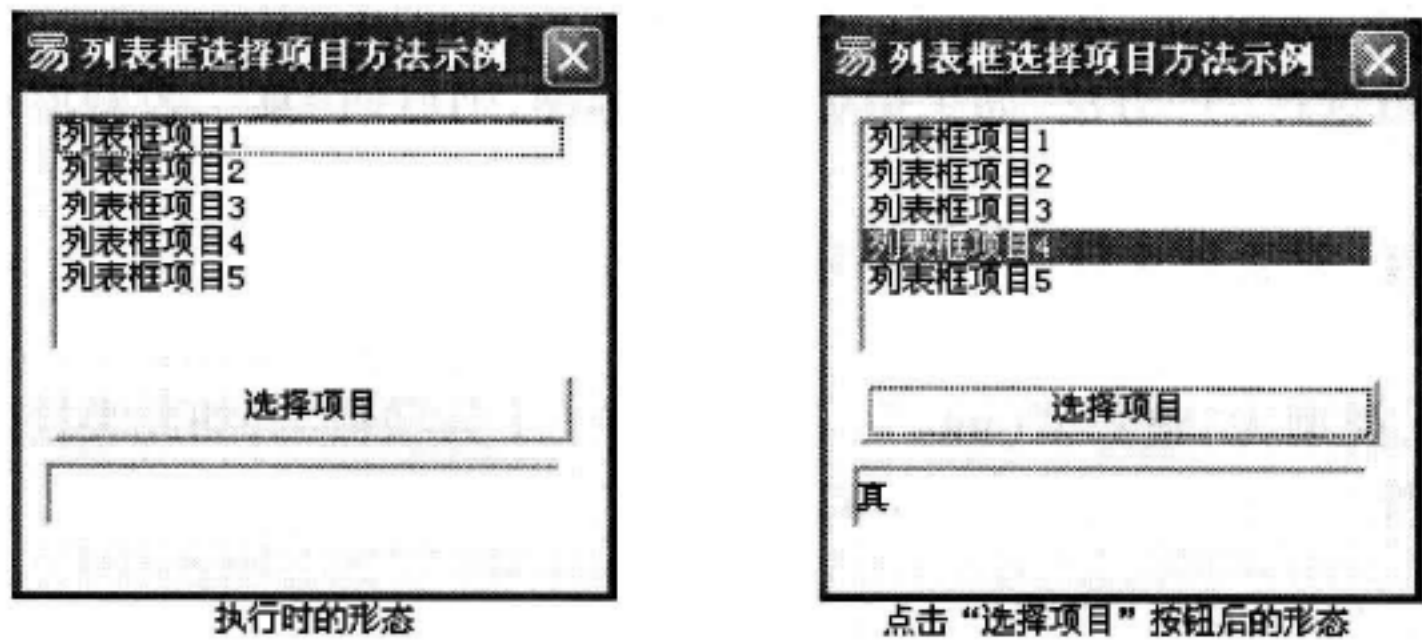



图 8-61 列表框“选择项目”示例

(11) “选择()”方法负责在所有项目中寻找首部包含指定文本的项目,如找到,则选中它,并返回该项目的位置索引,否则返回 -1。本命令仅在单选列表框中使用,如果在多选列表框中使用,将返回 -1。

调用格式:

〈整数型〉对象. 选择(欲选择的项目文本)

参数说明:

“欲选择的项目文本”,类型为“文本型(text)”。

【范例 8-13】用列表框设计实现选择指定列表框项目功能。具体参考例程 8-41。

列表框中原有 5 个项目,选择指定的列表项目,有就在第一个编辑框中返回“选中”,否则就返回“未选中”。如图 8-62 所示。

窗口程序集名	保留	保留	备注
窗口程序集1			

子程序名	返回值类型	公开	备注
_按钮1_被单击			

```
列表框1.允许选择多项 = 假
如果 (列表框1.选择 (“易语言”) ≠ -1)
    编辑框1.内容 = “选中”
    编辑框1.内容 = “未选中”
编辑框2.内容 = “有易语言就选(不允许多项的情况下)”
列表框1.允许选择多项 = 真
```

图 8-62 列表框“选择方法”代码示例

【运行结果】运行例程 8-41,观测单击“选择”按钮的变化,如图 8-63 所示。

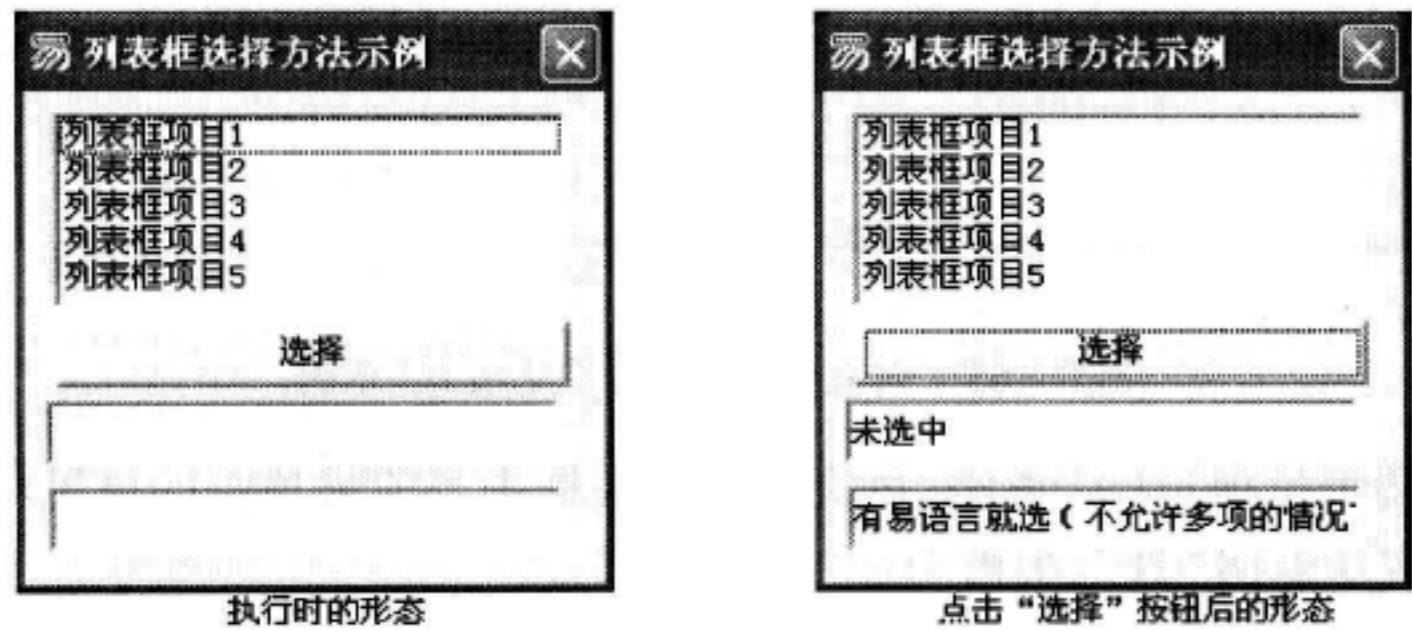


图 8-63 列表框“选择方法”示例

【注意】列表框“选择()”方法仅允许在单选列表框中使用。

(12) “是否被选择()”方法,如果指定项目被选择,则返回“真”,否则返回“假”。

调用格式:

〈逻辑型〉对象.是否被选择(项目索引)

参数说明:

“项目索引”,类型为“整数型(int)”。0 为项目一,1 为项目二,如此类推。

如图 8-64 所示,具体参考例程 8-42。

输入框(“请输入要查询的项目号:”,变量 1,)

编辑框 1. 内容 = 到文本(列表框 1. 是否被选择(到数值(变量 1)))

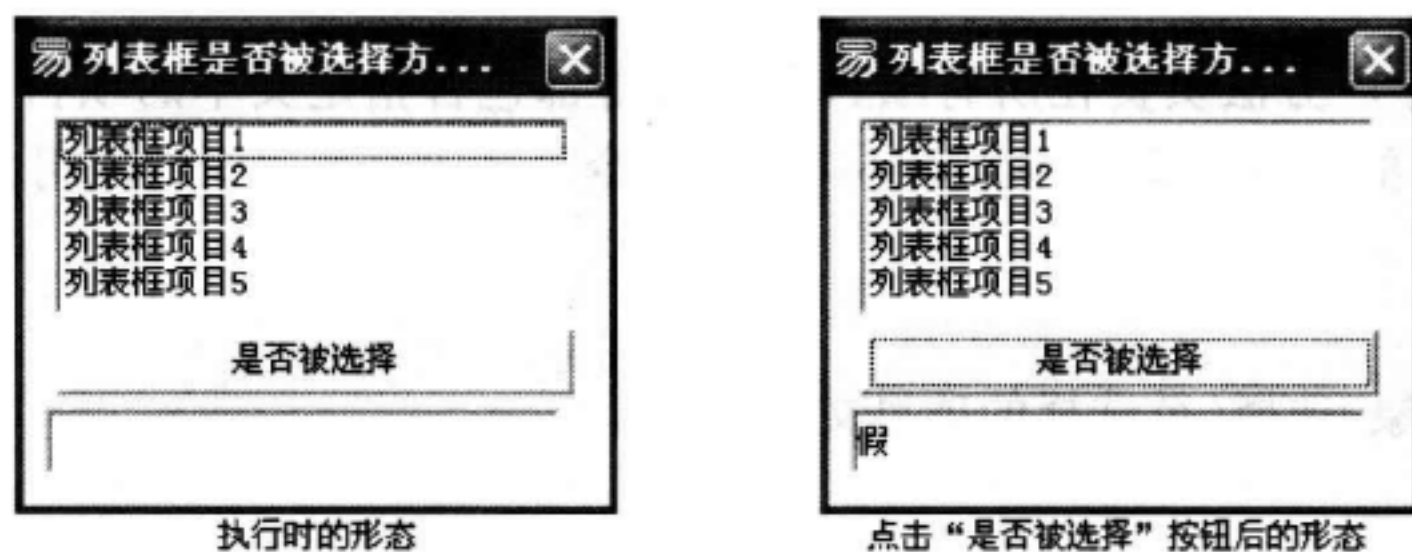


图 8-64 列表框“是否被选择”示例

(13) “取顶端可见项目()”方法,返回列表框中当前最顶端可见项目的索引。0 为项目一,1 为项目二,如此类推。失败返回 -1。

调用格式:

〈整数型〉对象.取顶端可见项目()

如图 8-65 所示,具体参考例程 8-43。

编辑框 1. 内容 = 到文本(列表框 1. 取顶端可见项目())

编辑框 2. 内容 = “当项目太多时,可以取最顶端的那个项目数值”

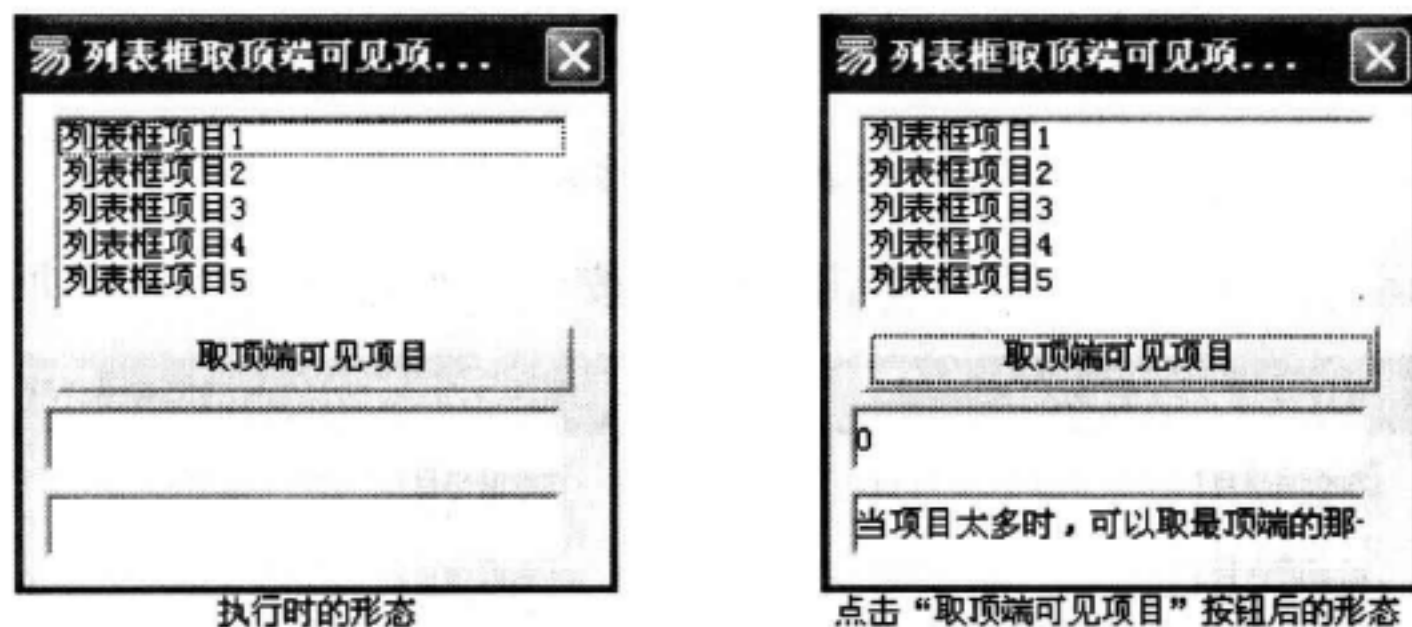


图 8-65 列表框“取顶端可见项目”示例

(14) “置顶端可见项目()”方法,负责设置列表框中当前最顶端的可见项目,必要时将自动滚动列表框。成功返回“真”,失败返回“假”。

调用格式:

〈逻辑型〉对象.置顶端可见项目(项目索引)

参数说明:

“项目索引”,类型为“整数型(int)”。0 为项目一,1 为项目二,如此类推。

如图 8-66 所示,具体参考例程 8-44。

列表框 1. 高度 = 80

列表框 1. 置顶端可见项目(3)

编辑框 1. 内容 = “当项目太多时,可以选择一个项目为必需可见的项目”

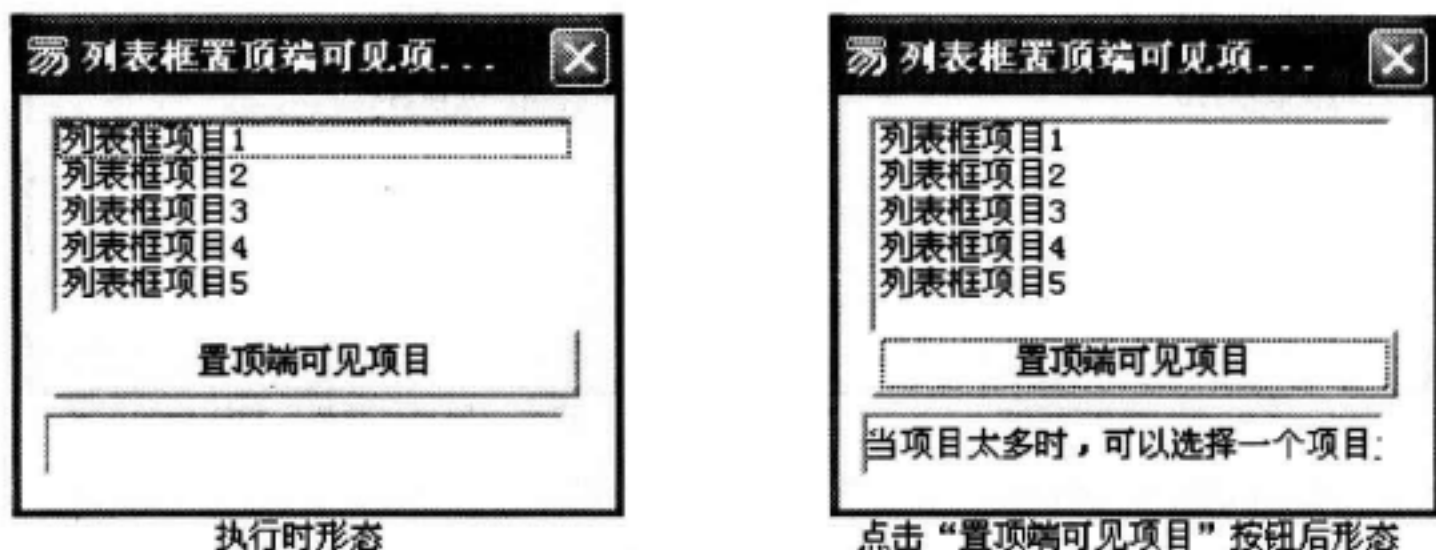


图 8-66 列表框“置顶端可见项目”示例

(15) “取项目数值()”方法,负责返回与指定项目相关联的数值。如果指定项目不存在,将返回 -1。

调用格式:

〈整数型〉对象. 取项目数值(项目索引)

参数说明:

“项目索引”,类型为“整数型(int)”。0 为项目一,1 为项目二,如此类推。

如图 8-67 所示,具体参考例程 8-45。

编辑框 2. 内容 = 到文本(列表框 1. 取项目数值(2))

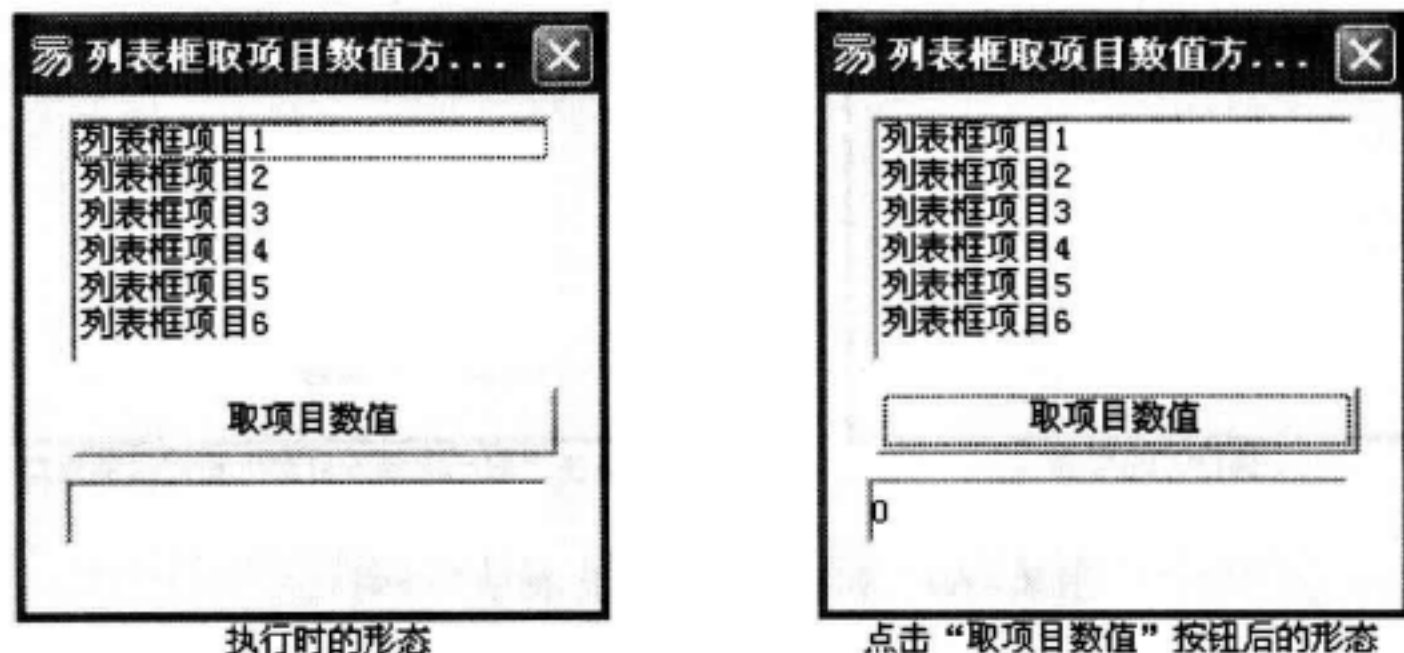


图 8-67 列表框“取项目数值”示例

(16) “置项目数值()”方法,负责设置与指定项目相关联的数值。成功返回“真”,失败返回“假”。

调用格式:

〈逻辑型〉对象. 置项目数值(项目索引,欲置入的项目数值)

参数说明:

“项目索引”,类型为“整数型(int)”。0 为项目一,1 为项目二,如此类推。

“欲置入的项目数值”,类型为“整数型(int)”。该项目数值与指定项目相关联。
如图 8-68 所示,具体参考例程 8-46。

编辑框 1. 内容 = “取项目文本为:” + 到文本(列表框 1. 置项目数值(2,4)) + “项目 3 值为 4”

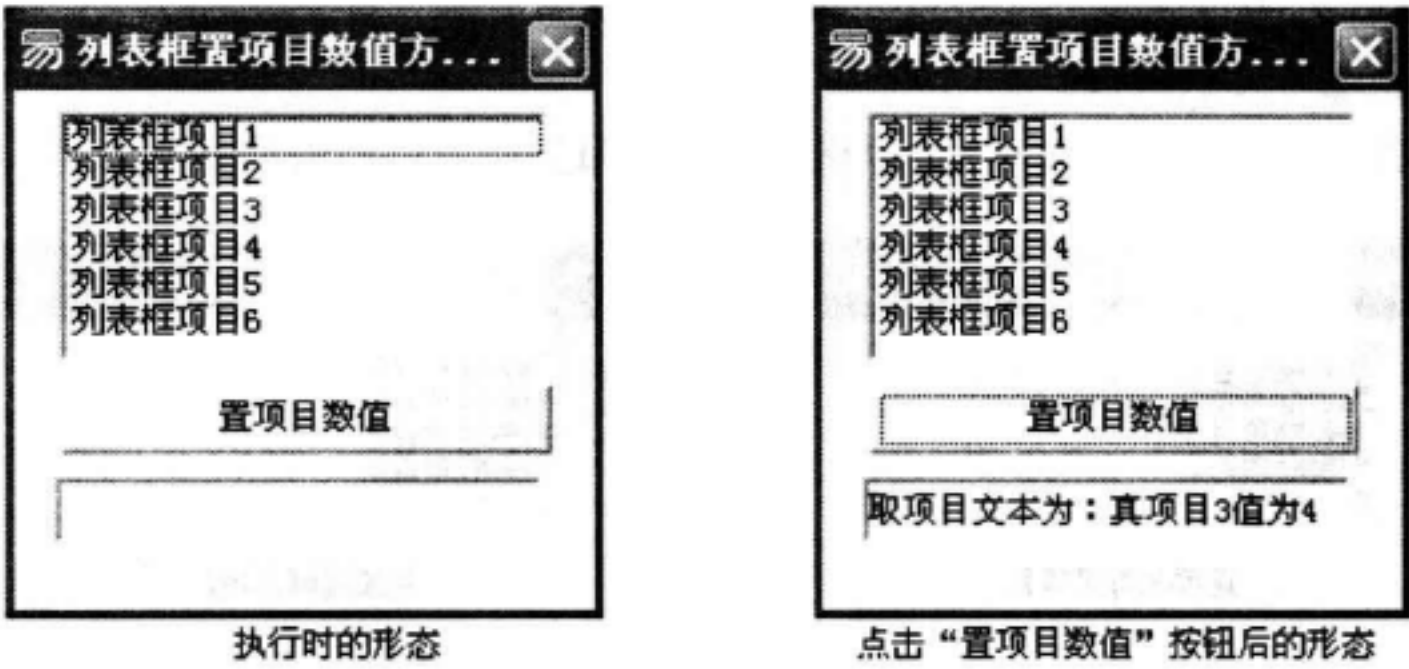


图 8-68 列表框“置项目数值”示例

(17) “取已选择项目数()”方法,负责返回已被选择项目的数目。
调用格式:

〈整数型〉对象.取已选择项目数()

如图 8-69 所示,具体参考例程 8-47。

编辑框 1. 内容 = 到文本(列表框 1. 取已选择项目数())
编辑框 2. 内容 = “已选择了” + 到文本(列表框 1. 取已选择项目数()) + “个项目”

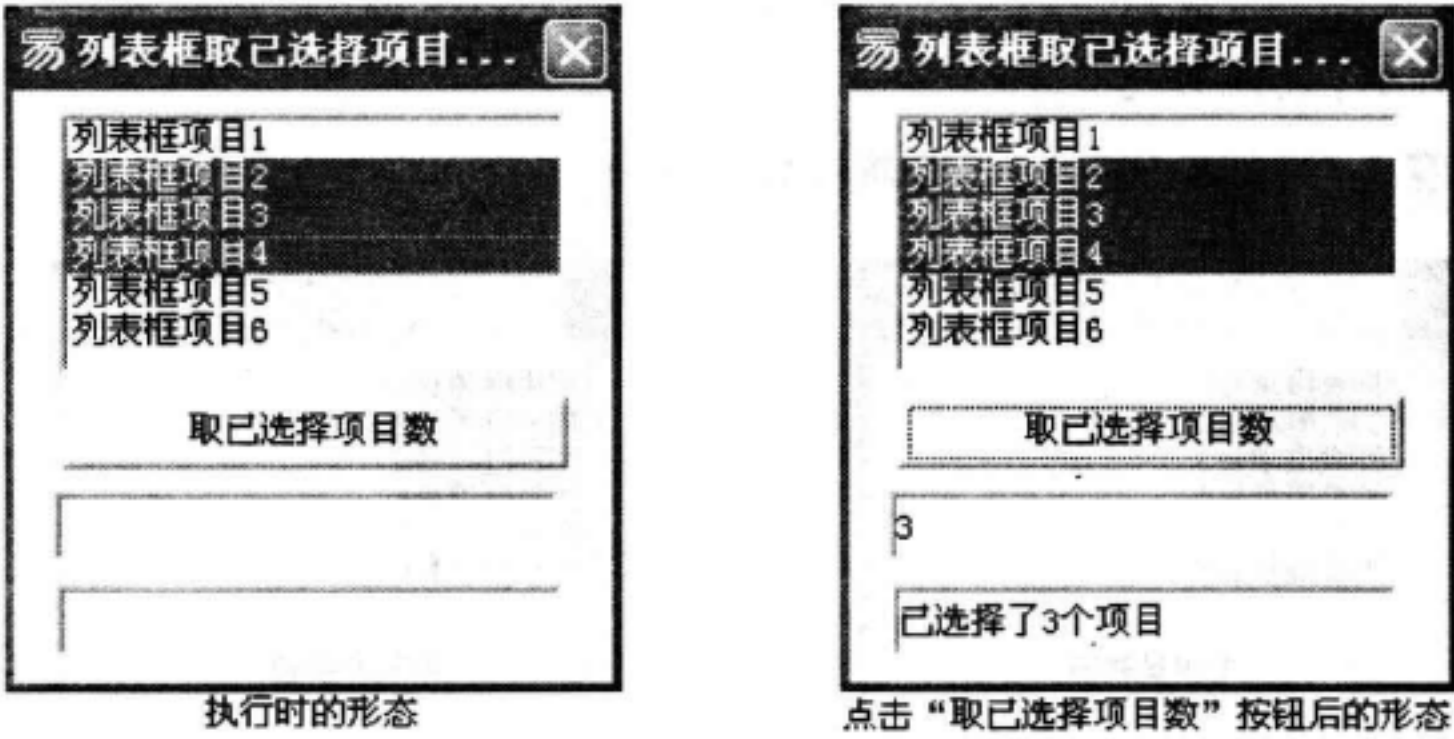


图 8-69 列表框“置项目数值”示例

(18) “取所有被选择项目()”方法,负责返回一个整数数组,内含所有当前被选择项目的位置索引。如果当前没有被选择项目,返回空数组。

调用格式:

〈整数型数组〉对象.取所有被选择项目()

【范例 8-14】用列表框设计实现取所有被选择项目列表框项目功能。具体参考例程 8-48。

列表框中原有 5 个项目,在编辑框中分别显示所有被选中的列表项目的索引和名称。具体的程序代码如图 8-70 所示。

窗口程序集名	保留	保留	备注
窗口程序集1			

子程序名	返回值类型	公开	备注
_按钮1_被单击			

变量名	类型	静态	数组	备注
变量	整数型		8	
计次	整数型			

```
编辑框1.内容 = ""
复制数组 (变量, 列表框1.取所有被选择项目 ()
计次循环首 (取数组成员数 (变量), 计次)
  编辑框1.内容 = 编辑框1.内容 + 到文本 (变量 [计次]) + "|"
计次循环尾 ()
编辑框1.内容 = 取文本左边 (编辑框1.内容, 取文本长度 (编辑框1.内容) - 1)
计次 = 1
编辑框2.内容 = ""
复制数组 (变量, 列表框1.取所有被选择项目 ()
计次循环首 (取数组成员数 (变量), 计次)
  编辑框2.加入文本 (到文本 (列表框1.取项目文本 (变量 [计次])) + "|")
计次循环尾 ()
```

图 8-70 列表框“取所有被选择项目”代码示例

【运行结果】运行例程 8-48, 观测单击“取所有被选择项目”按钮的变化, 如图 8-71 所示。

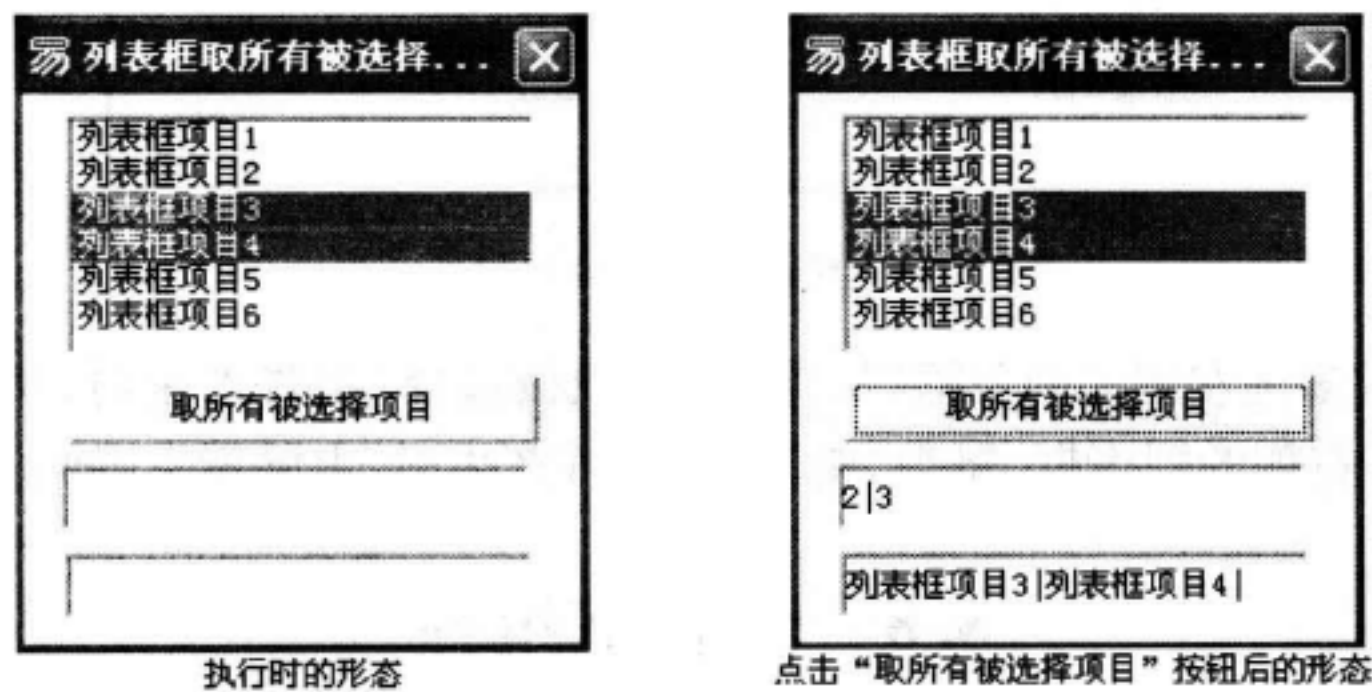


图 8-71 列表框“取所有被选择项目”示例

【注意】标题为“取所有被选择项目”的按钮被单击后, 如果列表框中有选择的项目, 那么就会在编辑框中显示被选择的项目文本。

8.7.4 列表框的事件

列表框的重要事件有“列表项被选择”、“双击选择”。当前所选择的列表项或区域被改变即产生“列表项被选择”事件。通常在此事件中读取列表框的“现行选中项”属性, 然后作相应操作。当用鼠标左键双击列表框时产生“双击”事件。下面通过一个简单的例子认识其功能。

【范例 8-15】用列表框设计实现列表项目被选择和被双击列表框事件功能。具体参考例程 8-49。

在两个列表框中均有 4 个项目, 选择第一个列表框中的项目时就会触发相关事件, 双击第二个列表框中的项目也会触发相关事件。如图 8-72 所示。

【运行结果】运行例程 8-49, 观测单击“取所有被选择项目”按钮的变化, 如图 8-73 所示。

窗口程序集名	保留	保留	备注
窗口程序集1			

子程序名	返回值类型	公开	备注
_列表框1_列表项被选择			

鸣叫 0

子程序名	返回值类型	公开	备注
_列表框2_双击选择			

鸣叫 0

子程序名	返回值类型	公开	备注
_标签1_反馈事件	整数型		
参数名	类型	参考	可空 数组 备注
参数一	整数型		
参数二	整数型		

图 8-72 列表框事件代码示例




图 8-73 列表框事件运行结果示例

【注意】单选或多选左侧的列表框,每当“列表项被选择”事件产生时,机箱喇叭或音箱就会鸣叫一声。双击右侧的列表框,每当“被双击”事件产生时,电脑就会鸣叫一声。

8.8 组合框组件

8.8.1 组合框概述

组合框组件可以将常用的一些选项呈列出来供挑选。它带有的下拉按钮收缩自如,可节约不少空间。组合框差不多是将编辑框和列表框特征组合在一起,既可以在控件的编辑框中输入数据,也可以在控件列表中选择项目。

组合框的属性方法事件等跟列表框基本相同,如使用加入项目方法来增加项目,使用删除项目方法来删除项目,使用清空方法来清除整个列表框,用现行选中项来返回或设置当前焦点所在的项目。两者虽然有很多相同点,但组合框有它特有的优势,组合框包含编辑区,可以输入列表框中不存在的选项。此外组合框节省窗体空间。

下面重点介绍组合框的属性、方法和事件。

8.8.2 组合框的属性

组合框的重要属性有“类型”、“列表项目”、“内容”、“现行选中项”、“去除重复”、“列表数值”、“自动排序”、“行间距”等。

“列表项目”属性中的“项目”,也就是组合框中的“行”,负责指定组合框中各项目文本。

此属性一般是必须设置的。也可以在程序运行时通过“加入项目”、“插入项目”、“删除项目”等方法维护组合框的列表项。该属性可以在设计期从弹出的“列表项管理对话框”中设置,如图 8-74 所示。


新建一个易语言程序时,在启动窗口中放置一个组合框组件,单击鼠标右键可以弹出“设置列表项目”菜单,或直接在属性面板的“列表项目”属性中单击按钮,也可以弹出相应的对话框,如图 8-75 所示。



图 8-74 列表项管理对话框



图 8-75 组合框“列表项目”

“类型”属性内容为整数型,有 3 个可选值:“0. 可编辑列表式”、“1. 可编辑下拉式”、“2. 不可编辑下拉式”,默认值为 1,可编辑下拉式。通常用到的是“可编辑下拉式”与“不可编辑下拉式”两种类型的组合框。“可编辑”表示允许用户向组合框输入文本,“不可编辑”则不允许,如图 8-76 所示,具体参考例程 8-50。

类型	可编辑列表式
内容	0. 可编辑列表式
最大文本长度	1. 可编辑下拉式
起始选择位置	2. 不可编辑下拉式
被选择字符数	0
被选择文本	** 设计时不可用
自动排序	假
行间距	1
文本颜色	黑色

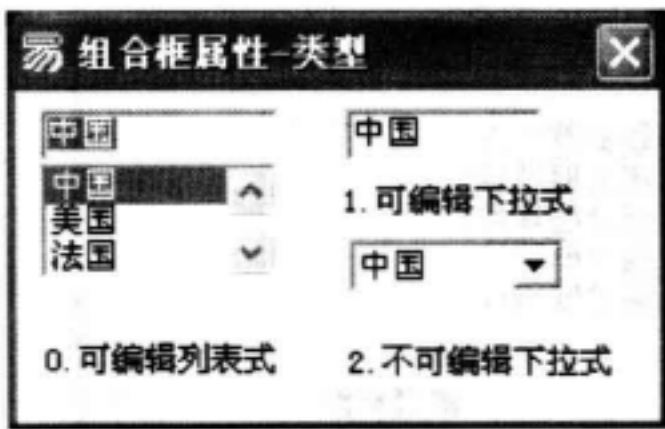


图 8-76 组合框“类型”

“内容”属性是组合框中的文本,类似于编辑框组件的“内容”属性。可以重复列表项目中的一个项目到“内容”属性,并且将现行选中项也设计为对应的数值。

“现行选中项”属性,表示当前被选中的项目的索引,整数型。如果本属性为 -1,则表示没有列表项被选中。组合框的第 1 个项目的索引为 0,第 2 个项目的索引为 1,以后累加 1,如此类推。

一般“内容”属性与“现行选中项”属性是配合起来使用的,否则会导致程序的混乱。

“去除重复”属性,易于理解,即当项目列表中的项目有重复的时候,可以将同名重复的合并为一个项目。“列表数值”、“自动排序”、“行间距”与列表框中的相关属性应用类似,这里就不再赘述了。

对于组合框组件,一般不需要特别复杂的操作,通常都需要设置“列表项目”和“现行选中项”属性,初学者要重点掌握。

8.8.3 组合框的方法

组合框的专有方法有“清空()”、“取项目数()”、“取项目文本()”、“置项目文本()”、“加入项目()”、“插入项目()”、“删除项目()”、“选择()”、“取顶端可见项目()”、“置顶端可见项目()”、“取项目数值()”、“置项目数值()”等。

(1) “清空()”方法,负责清除列表框中的所有项目。如图 8-77 所示,具体参考例程 8-51。

组合框 1. 清空()

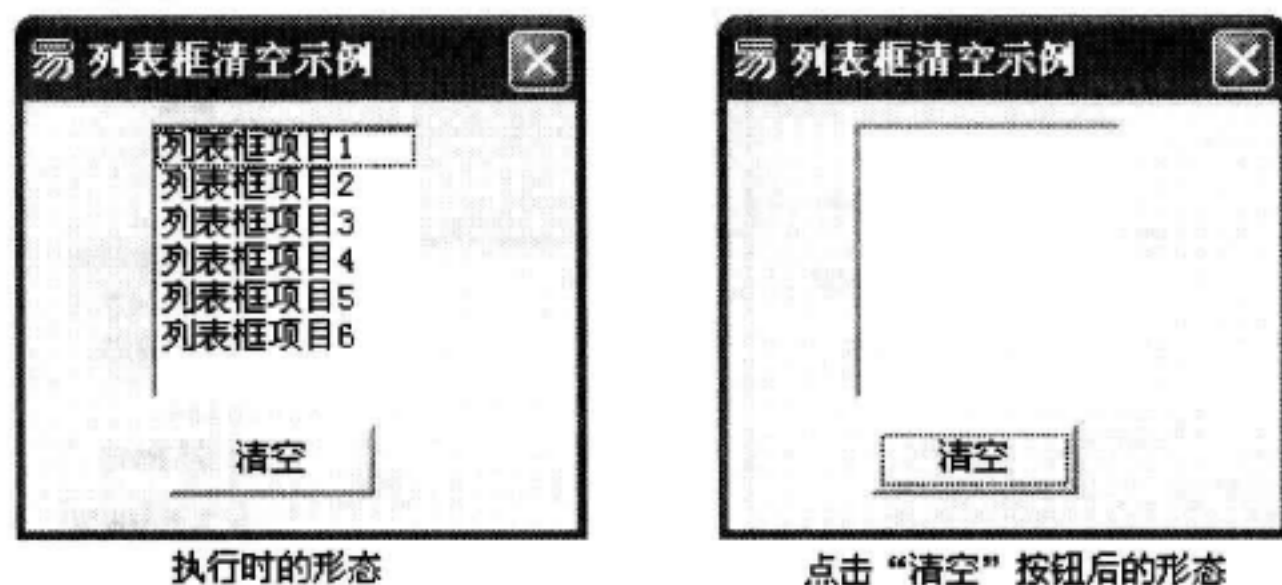


图 8-77 组合框“清空”示例

(2) “取项目数()”方法,负责返回当前项目的总数。如图 8-78 所示,具体参考例程 8-52。

编辑框 1. 内容 = “当前项目总数为:” + 到文本(组合框 1. 取项目数())

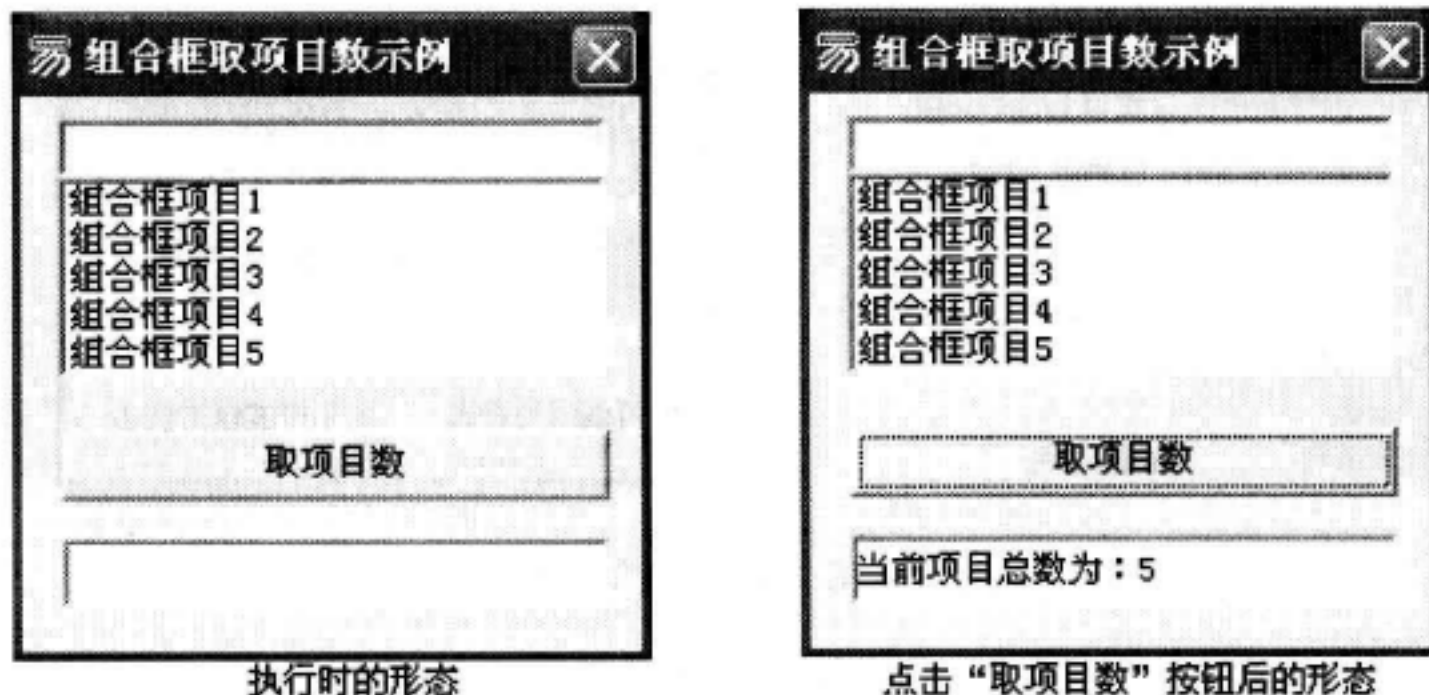


图 8-78 组合框“取项目数”示例

(3) “取项目文本()”方法,负责取指定的项目的文本。如图 8-79 所示,具体参考例程 8-53。

编辑框 1. 内容 = “取第三个项目的文本为:” + 到文本(组合框 1. 取项目文本(2))

(4) “置项目文本()”方法,负责设置指定的项目的文本。如图 8-80 所示,具体参考例程 8-54。



图 8-79 组合框“取项目文本”示例

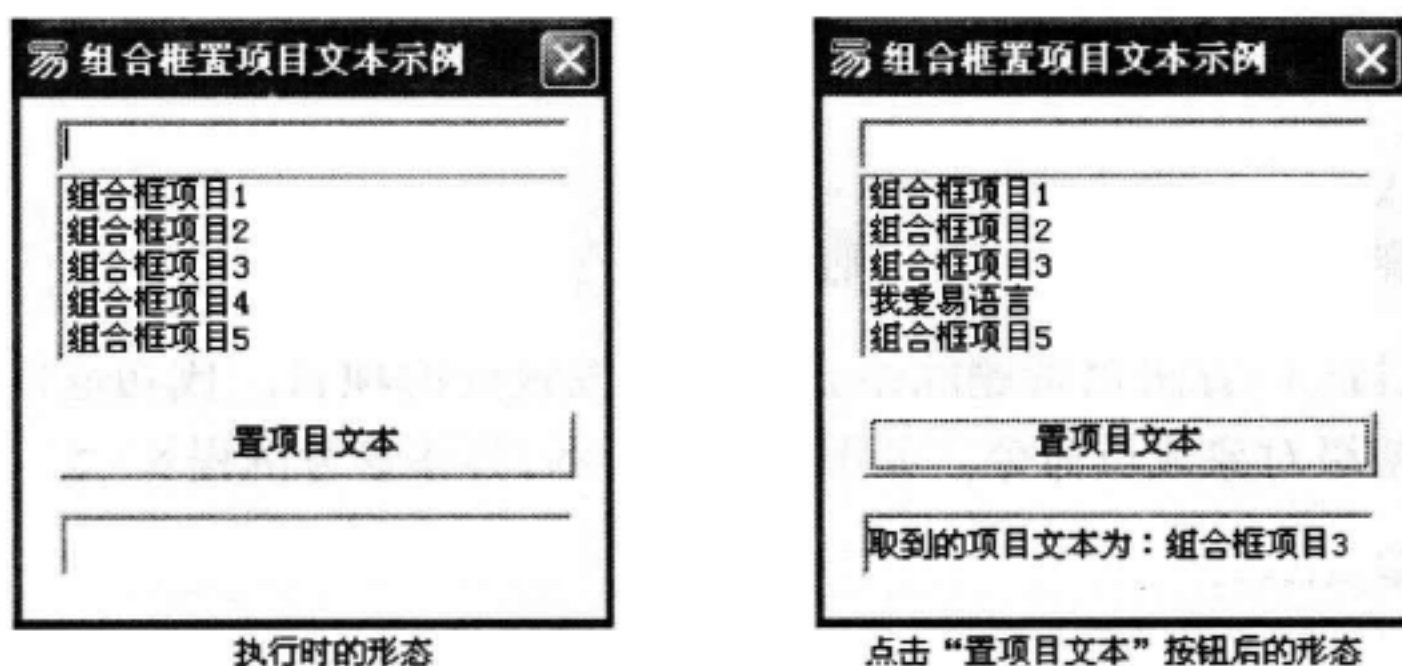


图 8-80 组合框“置项目文本”示例

组合框 1. 置项目文本(3,“我爱易语言”)

编辑框 1. 内容 = “将项目 4 改文本标题为:我爱易语言”

(5) “加入项目()”方法负责加入指定项目到组合框的尾部,成功返回加入后该项目所处的位置,失败返回 -1。如图 8-81 所示,具体参考例程 8-55。

组合框 1. 加入项目(“易语言”,)

编辑框 1. 内容 = “加入一个项目”

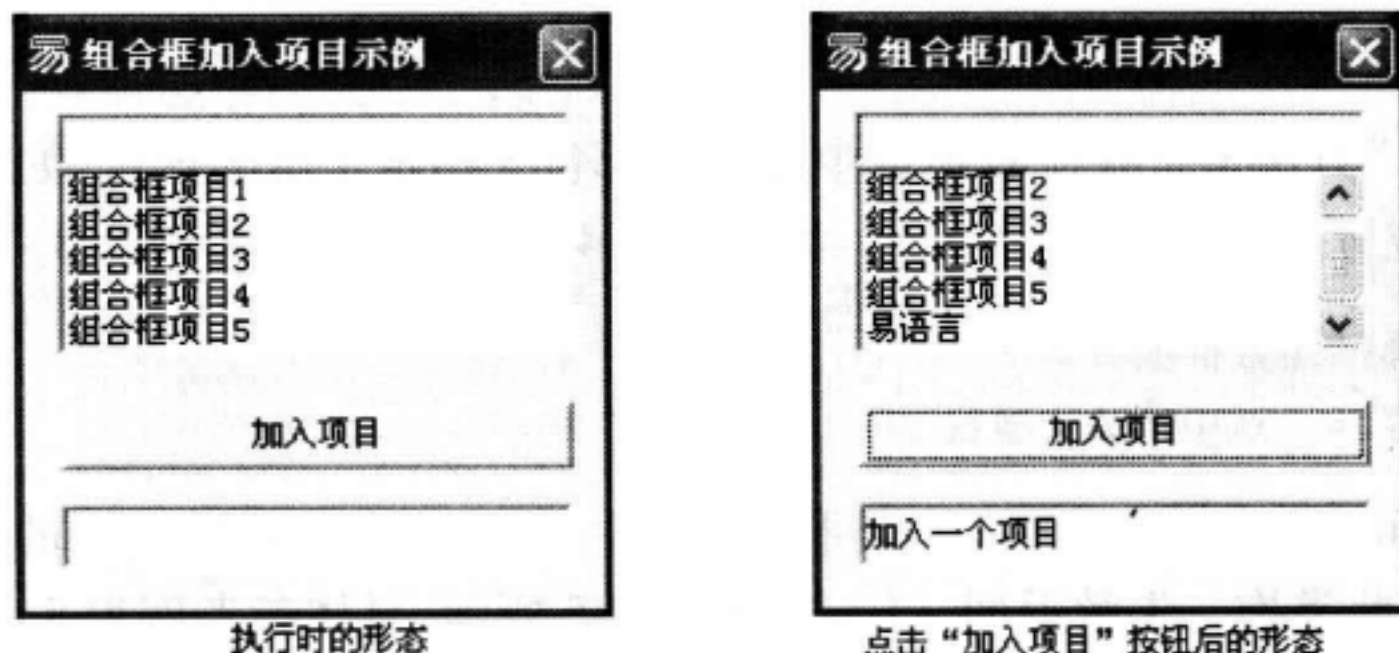


图 8-81 组合框“加入项目”示例

(6) “插入项目()”方法负责插入指定项目到组合框的指定位置处,成功返回插入后该项目所处的位置,失败返回 -1。本命令为初级对象成员命令。如图 8-82 所示,具体参考例程

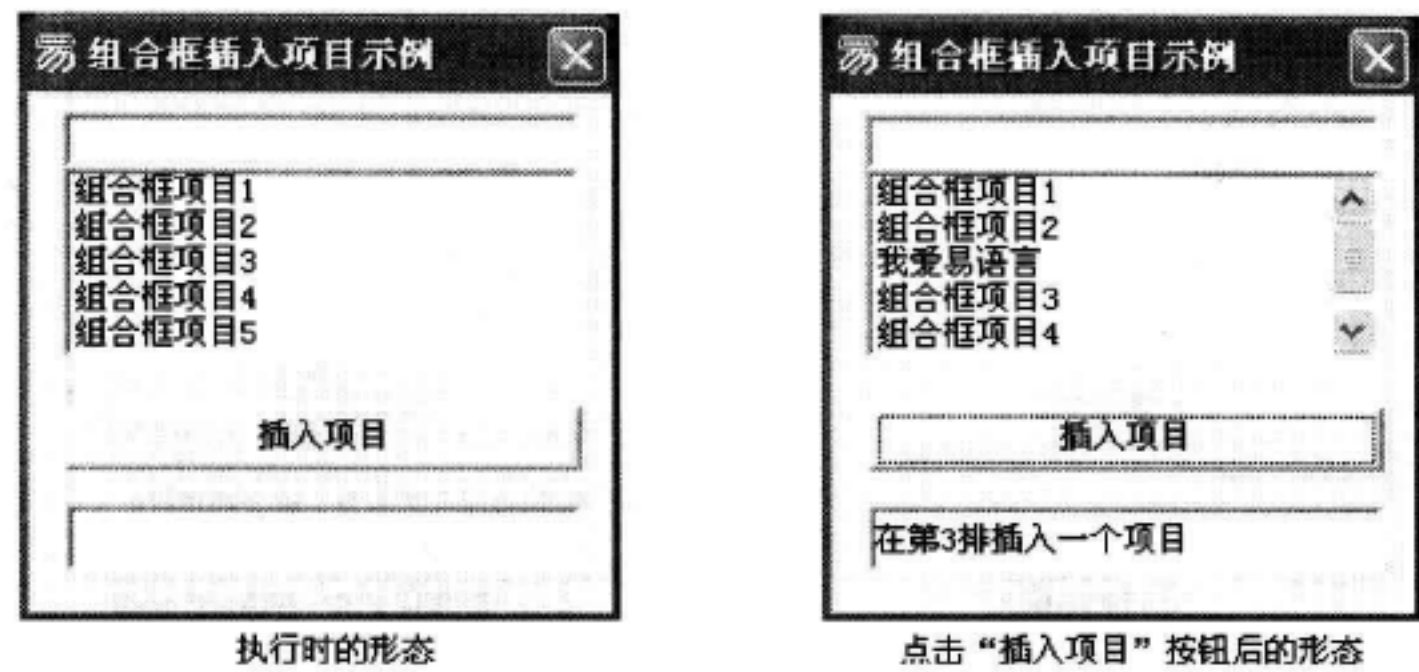


图 8-82 组合框“插入项目”示例

8-56。

组合框 1. 插入项目(2,“我爱易语言”,)
编辑框 1. 内容 = “在第 3 排插入一个项目”

(7) “删除项目()”方法负责删除组合框指定位置处的项目。成功返回“真”，失败返回“假”。本命令为初级对象成员命令。如图 8-83 所示,具体参考例程 8-57。

组合框 1. 删除项目(1)
编辑框 1. 内容 = “删除第 2 个项目”

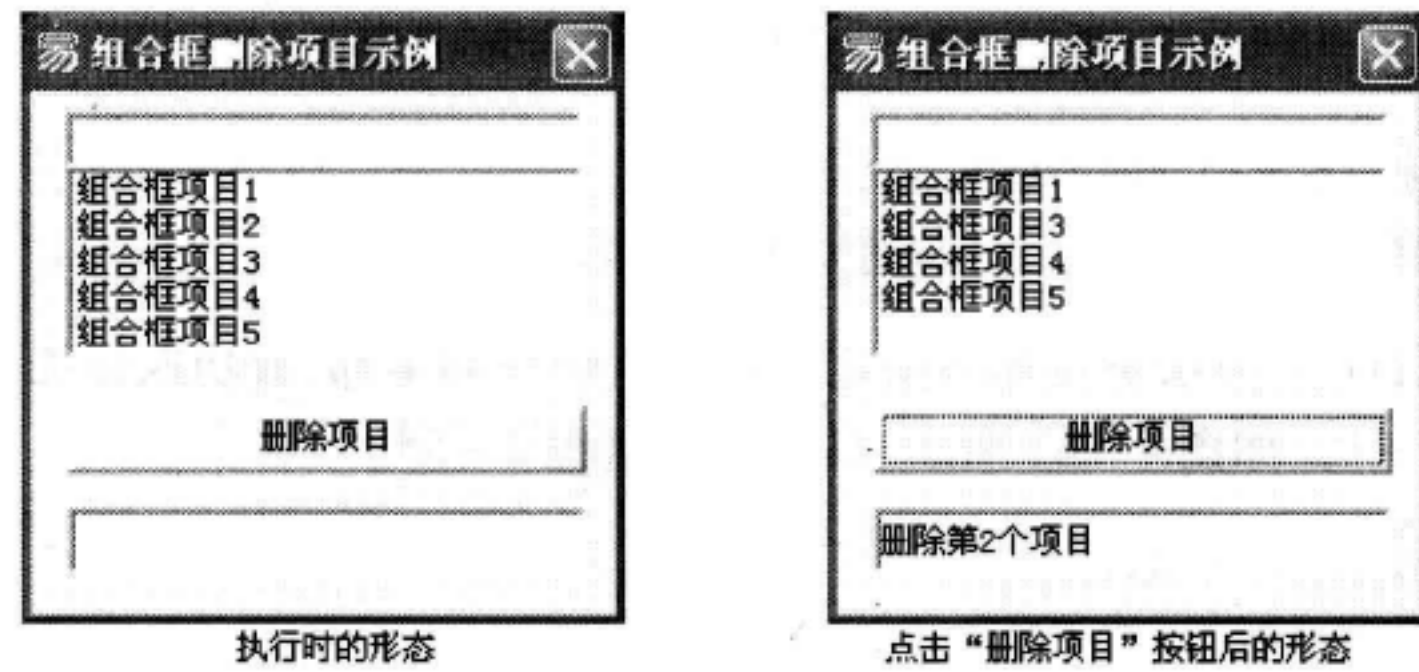


图 8-83 组合框“删除项目”示例

(8) “选择()”方法负责在所有项目中寻找首部包含指定文本的项目,如找到,则选中它,并返回该项目的位置索引,否则返回-1。如图 8-84 所示,具体参考例程 8-58。

组合框 1. 选择(“组合框项目 2”)
编辑框 1. 内容 = “选中第 2 个项目”

(9) “取顶端可见项目()”方法,返回组合框中当前最顶端可见项目的索引。0 为项目一,1 为项目二,如此类推。失败返回-1。如图 8-85 所示,具体参考例程 8-59。

编辑框 1. 内容 = 到文本(组合框 1. 取顶端可见项目())
编辑框 2. 内容 = “当项目太多时,可以取最顶端的那个项目数值”

(10) “置顶端可见项目()”方法,负责设置组合框中当前最顶端的可见项目,必要时将自动滚动组合框。成功返回“真”,失败返回“假”。如图 8-86 所示,具体参考例程 8-60。

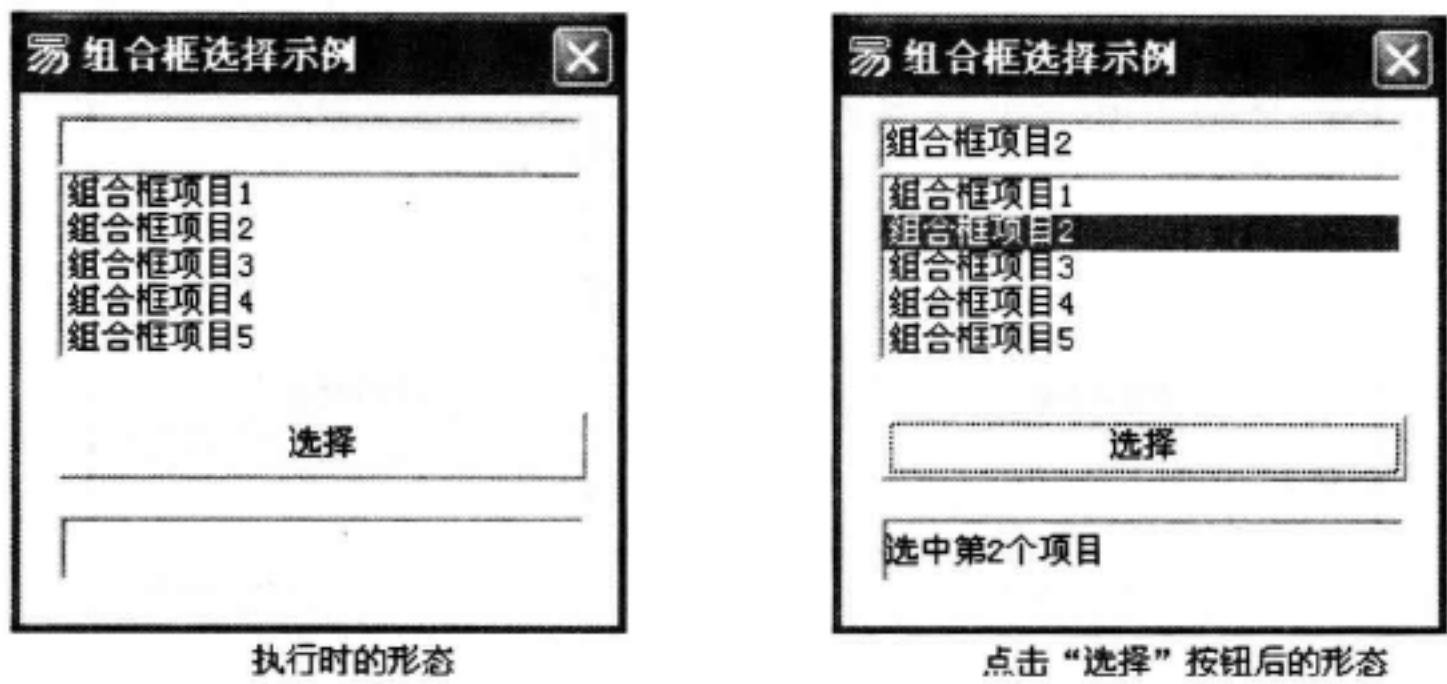


图 8-84 组合框“选择方法”示例

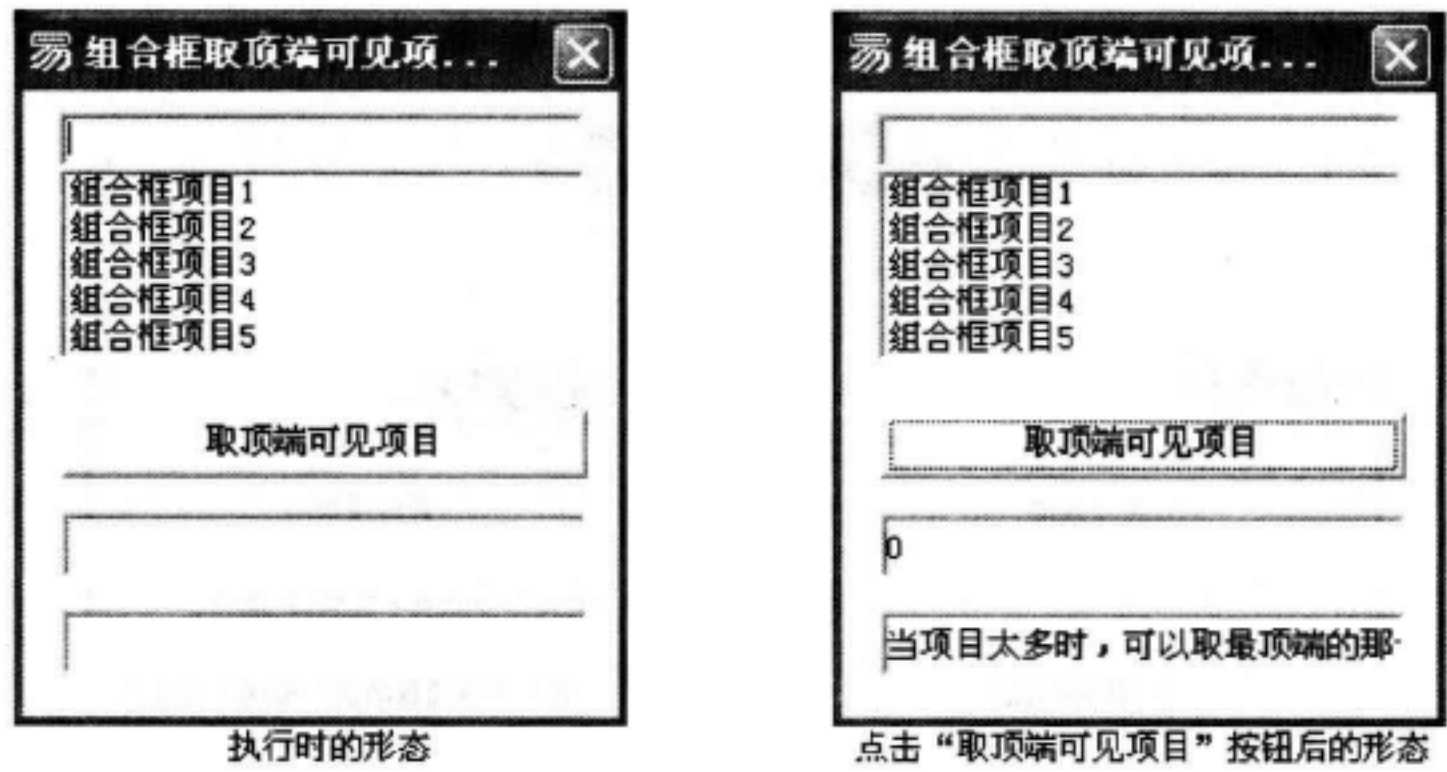


图 8-85 组合框“取顶端可见项目”示例

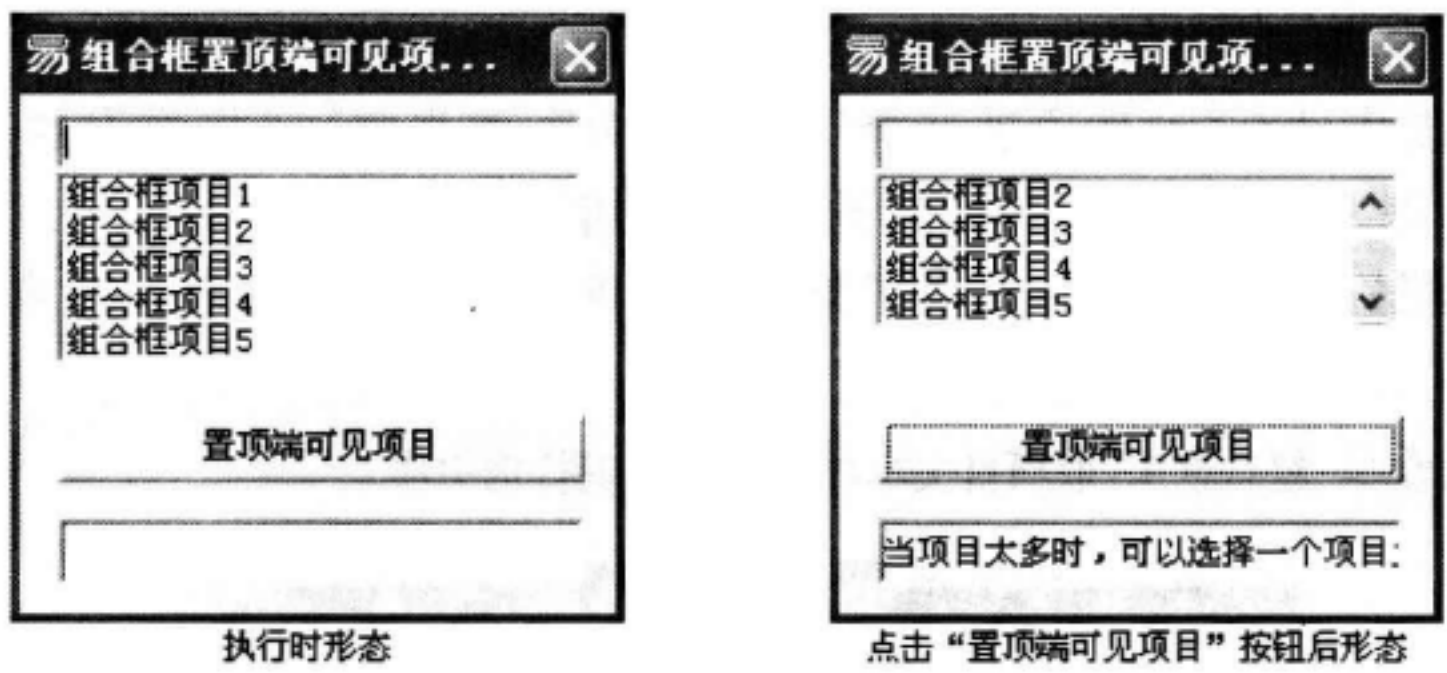


图 8-86 组合框“置顶端可见项目”示例

组合框 1. 高度 = 80
组合框 1. 置顶端可见项目(3)
编辑框 1. 内容 = “当项目太多时,可以选择一个项目为必需可见的项目”

(11) “取项目数值()”方法,负责返回与指定项目相关联的数值。如果指定项目不存在,将返回 -1。如图 8-87 所示,具体参考例程 8-61。

编辑框 1. 内容 = 到文本(组合框 1. 取项目数值(2))

(12) “置项目数值()”方法,负责设置与指定项目相关联的数值。成功返回“真”,失败返

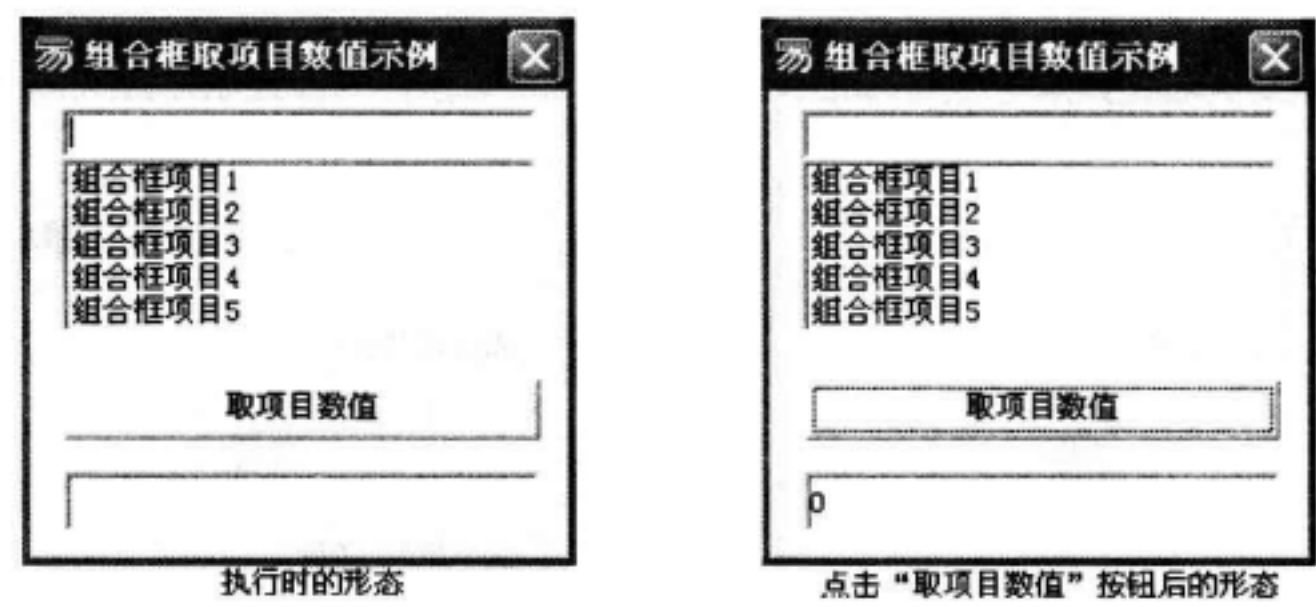


图 8-87 组合框“取项目数值”示例

回“假”。如图 8-88 所示,具体参考例程 8-62。

编辑框 1. 内容 = “取项目文本为:” + 到文本(组合框 1. 置项目数值(2,4)) + “项目 3 值为 4”

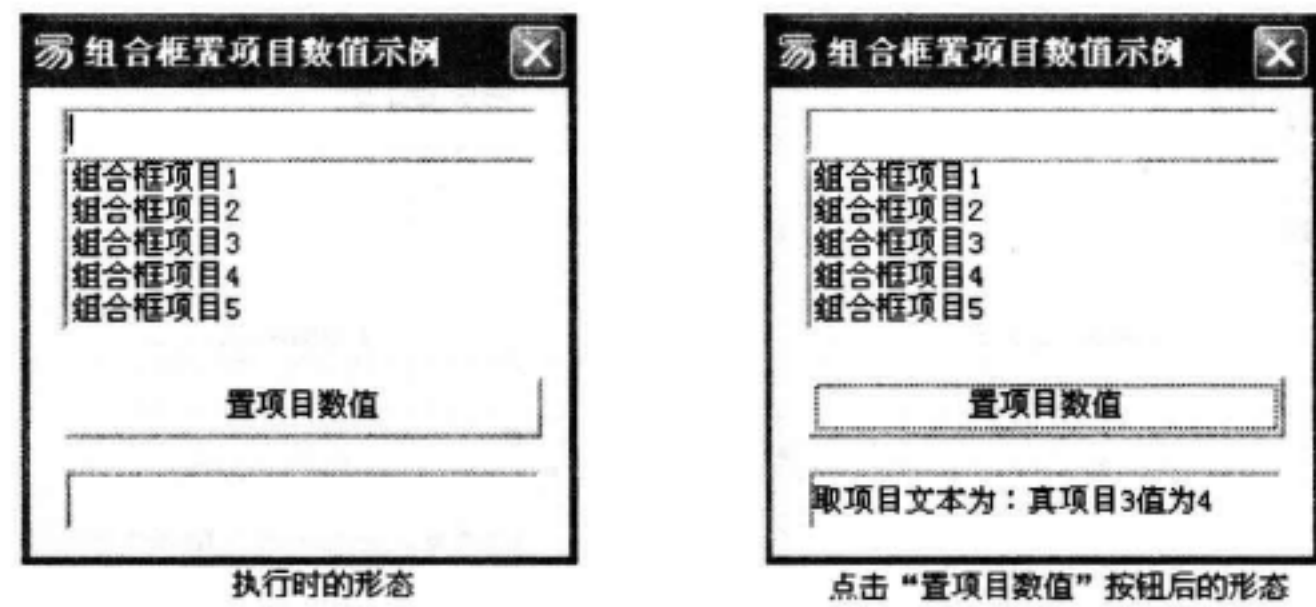


图 8-88 组合框“置项目数值”示例

8.8.4 组合框的事件

组合框的重要事件有“列表项被选择”、“编辑内容被改变”、“将弹出列表”、“列表被关闭”、“双击选择”等。下面通过简单的例子认识其功能。

(1) “列表项被选择”事件,在选择了某个列表项时产生此事件。如图 8-89 所示,具体参考例程 8-63。

编辑框 1. 内容 = 组合框 1. 取项目文本(组合框 1. 现行选中项)

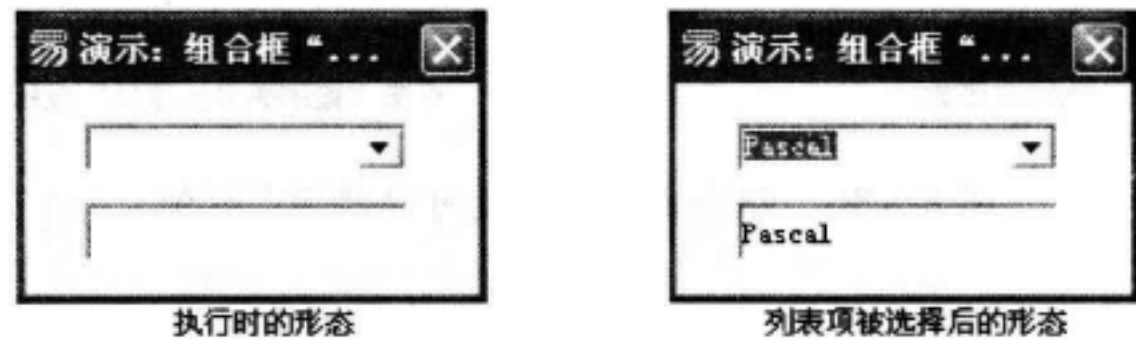


图 8-89 组合框“列表项被选择”示例

(2) “编辑内容被改变”事件,在用户向组合框中输入文本时产生此事件。

【范例 8-16】用组合框设计实现编辑内容被改变事件功能。具体参考例程 8-64。

在组合框中输入文字后,按钮的禁止属性才为“假”,并且可以将组合框中的文字通过单击此按钮追加到组合框中去。具体的程序代码如图 8-90 所示。

【运行结果】运行例程 8-64,观测单击“追加项目”按钮的变化,如图 8-91 所示。

窗口程序集名	保留	保留	备注
窗口程序集1			

子程序名	返回值类型	公开	备注
组合框1_编辑内容被改变			

按钮1.禁止 = 假

子程序名	返回值类型	公开	备注
按钮1_被单击			

组合框1.加入项目 (组合框1.内容,)

图 8-90 “编辑内容被改变”代码示例

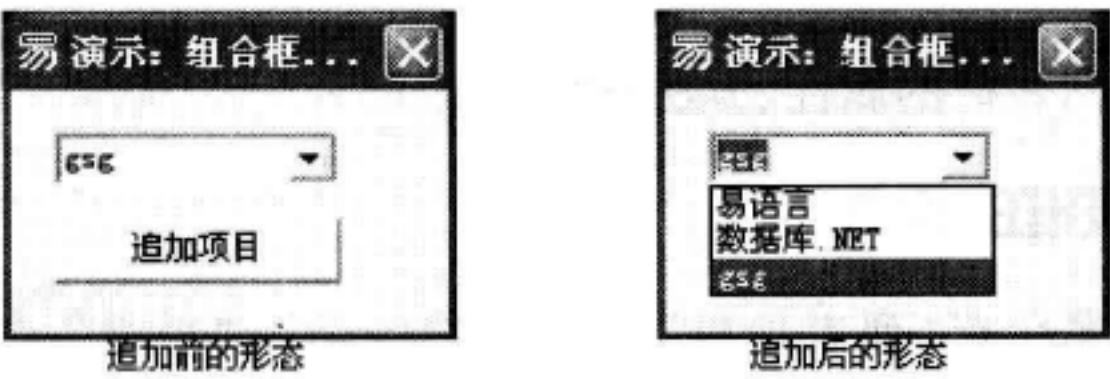


图 8-91 “编辑内容被改变”示例

【注意】首要条件是,组合框的类型属性为“可编辑下拉式”或“可编辑列表式”。

- (3) “将弹出列表”事件,在组合框的下拉列表部分即将弹出之前产生此事件。
- (4) “列表被关闭”事件,在组合框的下拉列表部分被关闭后产生此事件。
- (5) “双击选择”事件,在用户双击了组合框的某列表项时产生此事件。

对于组合框组件,一般不需要特别复杂的操作,通常都需要设置“列表项目”和“现行选中项”属性。

【范例 8-17】用组合框实现问答调查。具体参考例程 8-65。

可以选择组合框中项目,也可以在组合框中添加新的内容,点击“提交”按钮后实现相应选项的提示功能。如图 8-92 所示。

窗口程序集名	保留	保留	备注
窗口程序集1			

子程序名	返回值类型	公开	备注
按钮1_被单击			

信息框 (标签1.标题 + “ ” + 组合框1.内容, 0,)

对于类型为“可编辑下拉式”的组合框,只能用“内容”属性来读取组合框中的值。

(因为组合框中的内容有可能被用户修改,所以不能这样读取其内容:

“组合框1.取项目文本(组合框1.现行选中项)”)

图 8-92 组合框范例代码示例

【运行结果】运行例程 8-65,观测单击“提交”按钮的变化,如图 8-93 所示。

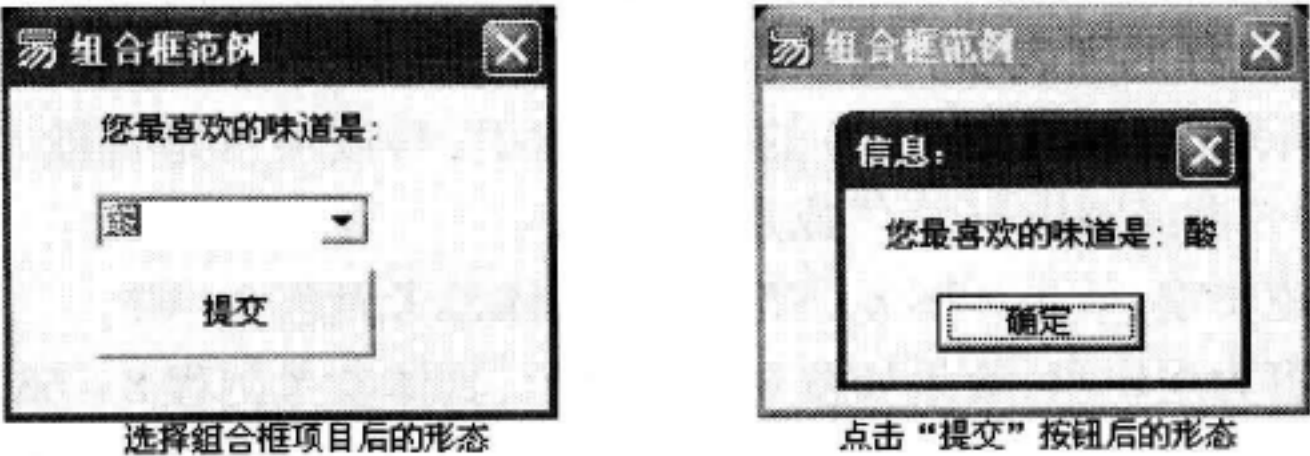



图 8-93 组合框范例示例

【注意】组合框的类型为“可编辑下拉式”时,才能用“内容”属性来读取组合框中的值。

8.9 选择列表框组件

8.9.1 选择列表框概述

选择列表框组件类似选择框与单选框,可以有多选或单选的性质。选择列表框既像选择框,又类似列表框,也能将它看作是一个有选择功能的列表框,可以非常快速地批量加入项目。

下面重点介绍选择列表框的属性、方法和事件。

8.9.2 选择列表框的属性

选择列表框的重要属性有“列表项目”、“现行选中项”、“单选”、“多列”、“行间距”、“自动排序”、“项目数值”等。


“列表项目”属性用于在界面设计阶段设置选择列表框的原始项目。可以在选择列表框上用鼠标右键弹出“设置列表项目”菜单,或直接在属性面板的“列表项目”属性中单击钮,也可以弹出相应的对话框,如图 8-94 所示,具体参考例程 8-66。



图 8-94 列表项管理对话框

“现行选中项”属性,负责指定列表框中现行拥有焦点的项目的索引。索引值从 0 开始,-1 表示现行没有被选择的列表项。可在设计期设置或在运行期设置和读取该属性的值。本属性仅指定现行拥有焦点(被高亮显示)的项目的索引,与该项目是否被“被选中”无关(“被选中”即在项目前的方框或圆中打“√”或点“·”)。

“单选”内容为逻辑型,效果如图 8-95 所示,具体参考例程 8-67。

“多列”属性内容为逻辑型,效果如图 8-96 所示,具体参考例程 8-68。

“行间距”、“自动排序”、“项目数值”与列表框中的相关属性应用类似,这里就不再赘述了。

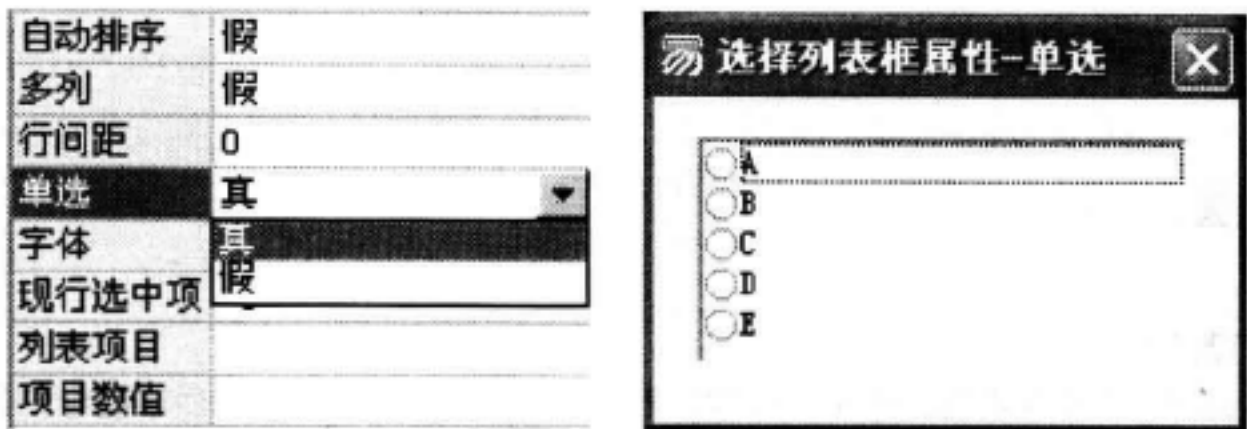


图 8-95 选择列表框“单选”

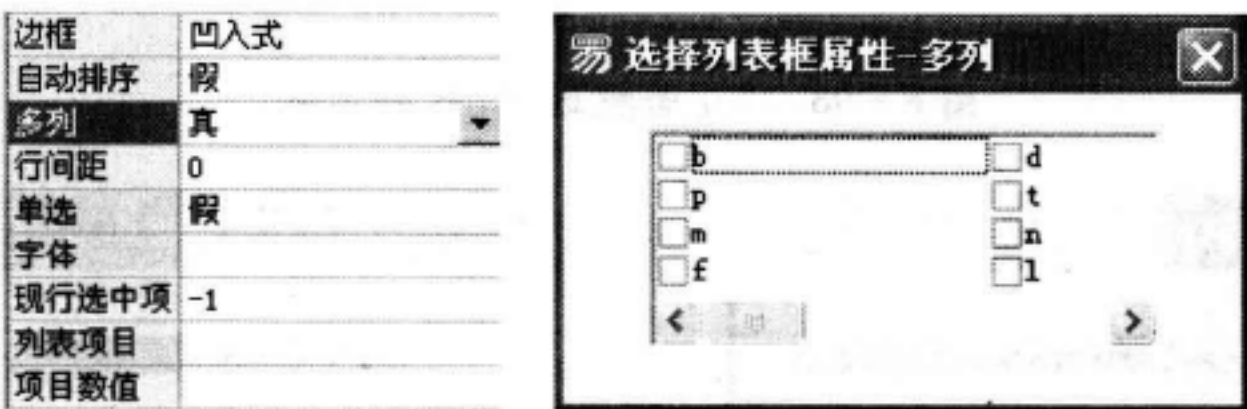


图 8-96 选择列表框“多列”

8.9.3 选择列表框的方法

选择列表框的重要方法有“选中项目()”、“是否被选中()”、“允许()”、“是否被允许()”等,其他方法与列表框组件的同名方法类似。

(1) “选中项目()”方法,负责使选择列表框中的某一项目(某一行)被选中(打“√”或“·”)。第 1 个参数指定要进行操作的项目的索引(从 0 开始,即第一个项目的索引是 0,依次类推);第 2 个参数是逻辑型值,指定是“选中”(真)还是“取消选中”(假)。第 2 个参数可以被省略,省略时默认为“真”。如图 8-97 所示,具体参考例程 8-69。

选择列表框 1. 选中项目(0,真)

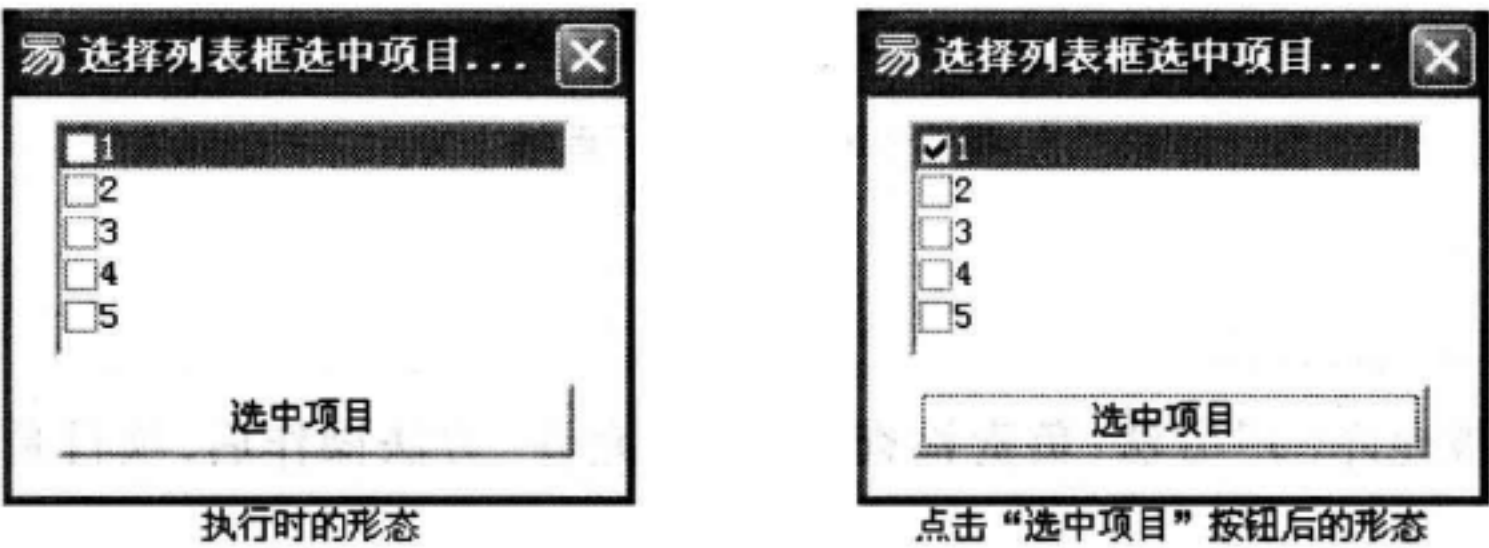


图 8-97 选择列表框“选中项目”示例

(2) “是否被选中()”方法,负责判断某一项目是否被选中,如果被选中返回“真”,否则返回“假”。

【范例 8-18】用选择列表框设计实现是否被选中方法功能。具体参考例程 8-70。

在选择列表框中选中指定项目,点击按钮后取消选中状态,否则提示先选中此选项。如图 8-98 所示。

【运行结果】运行例程 8-70,观测单击“是否被选中”按钮的变化,如图 8-99 所示。

【注意】指定要进行操作的项目的索引(从 0 开始,即第 1 个项目的索引是 0,依次类推)。

(3) “允许()”方法,负责将一个选择列表框的第 N 项设置为不可选。如图 8-100 所示,

窗口程序集名	保留	保留	备注
窗口程序集1			

子程序名	返回值类型	公开	备注
_按钮1_被单击			

如果真 (选择列表框1.是否被选中 (1) = 假)
 信息框 (“请先选中 选择列表框1 的第二个项目!” + #换行符 + “然后才能看到效果。” , 0,)
选择列表框1.选中项目 (1, 假) ‘取消选中第二个项目
’ 执行这一行代码前, 应先选中第二个项目, 否则看不到效果。

图 8-98 “是否被选中”代码示例

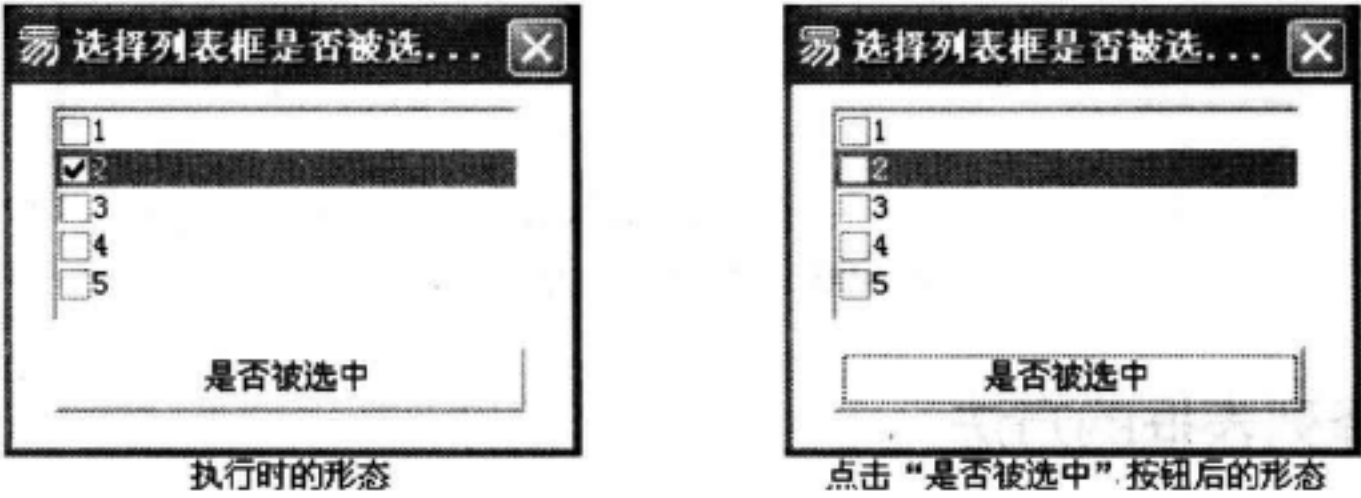


图 8-99 “是否被选中”示例

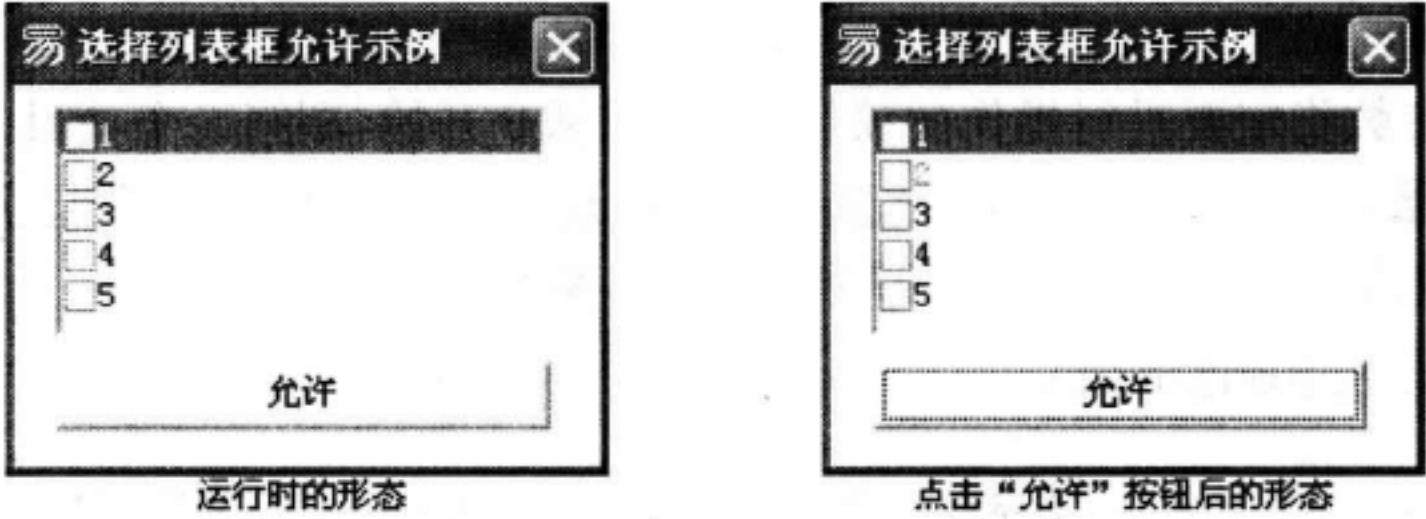


图 8-100 “允许”示例

具体参考例程 8-71。

选择列表框 1. 允许(1,假)

(4) “是否被允许()”方法,负责检查上述用“允许”方法操作后,项目是否被允许的状态。如图 8-101 所示,具体参考例程 8-72。

信息框(“是否允许:” + 到文本(选择列表框 1. 是否被允许(1)),0,)

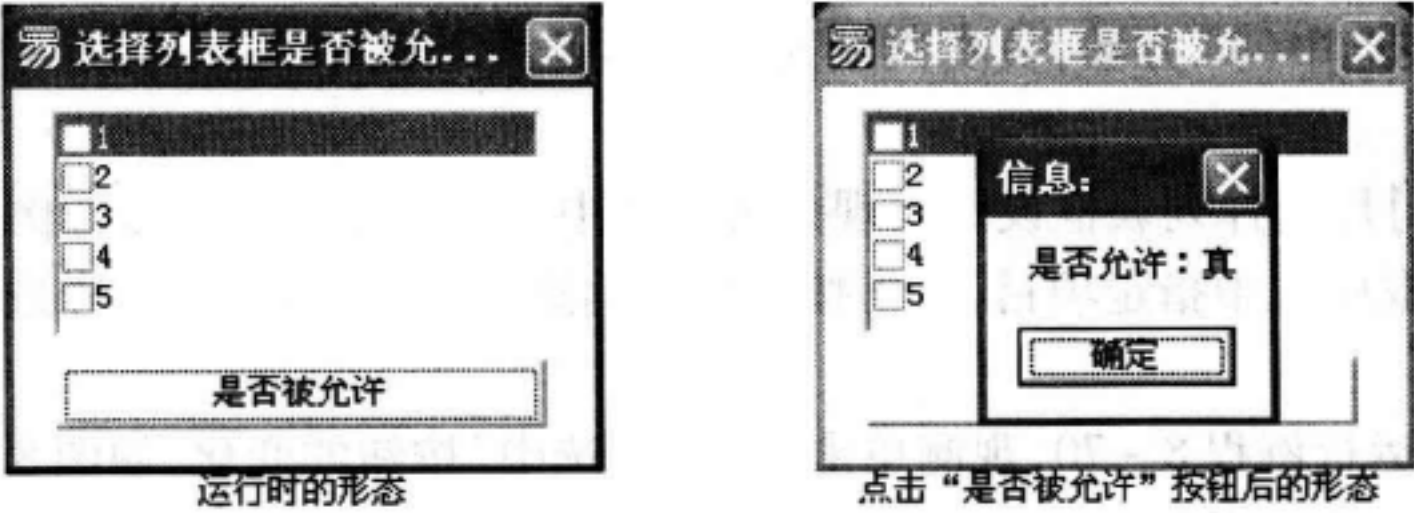


图 8-101 “是否被允许”示例

8.9.4 选择列表框的事件

选择列表框的重要事件有“选中状态被改变”、“列表项被选择”、“被双击”。下面通过简单的例子认识其功能。

“选中状态被改变”事件：当单击某项目使其状态由“选中”变为“取消选中”或“由取消选中”变为“选中”时产生此事件。在程序运行时用代码改变项目的状态不会引发此事件。如图 8-102 所示，具体参考例程 8-73。



图 8-102 选择列表框“选中状态被改变”示例

【范例 8-19】用选择列表框设计实现是否被选中方法功能。具体参考例程 8-74。

在选择列表框中选中项目 1，点击选中项目 1 取消选中状态后，提示“您取消选中了选择列表框的第 1 项”。如图 8-103 所示。

窗口程序集名	保留	保留	备注
窗口程序集1			

子程序名	返回值类型	公开	备注
_选择列表框_选中状态被改变			

变量名	类型	静态	数组	备注
x	逻辑型			

x = 选择列表框.是否被选中 (选择列表框.现行选中项)

如果 (x = 真)

 信息框 (“您选中了 选择列表框的第 ” + 到文本 (选择列表框.现行选中项 + 1) + “ 项。” , 0,)

 信息框 (“您取消选中了 选择列表框的第 ” + 到文本 (选择列表框.现行选中项 + 1) + “ 项。” , 0,)

“现行选中项”从0开始计，即0为第一项，1为第二项……

图 8-103 “编辑内容被改变”代码示例

【运行结果】运行例程 8-74，观测单击“是否被选中”按钮的变化，如图 8-104 所示。

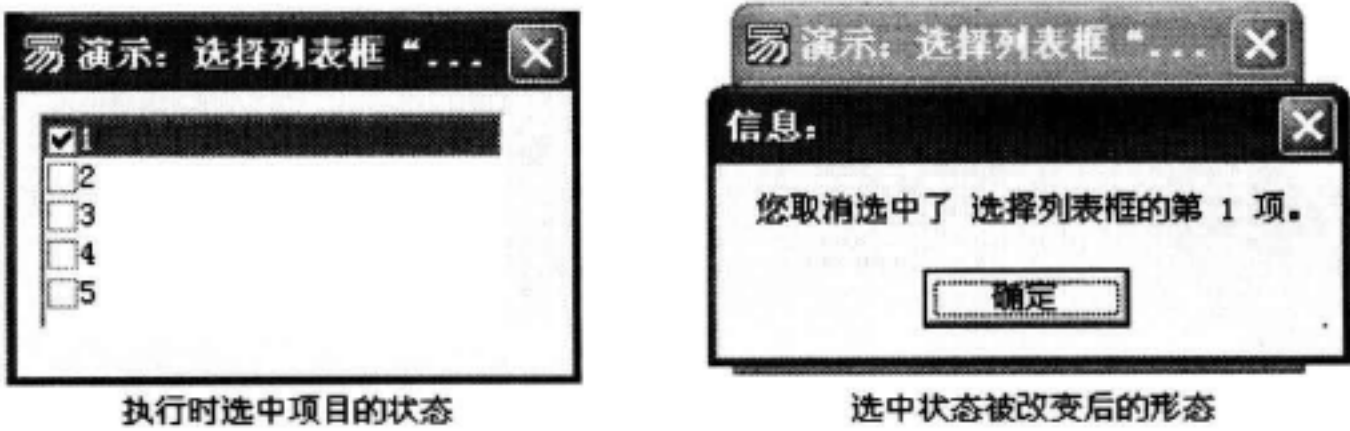


图 8-104 “编辑内容被改变”运行结果示例

【注意】在程序运行时用代码改变项目的状态不会引发此事件。

(1) “列表项被选择”事件，如果当前所选择的列表项被改变即产生此事件。如图 8-105 所示，具体参考例程 8-75。

信息框 (“_选择列表框_列表项被选择”, 0,)

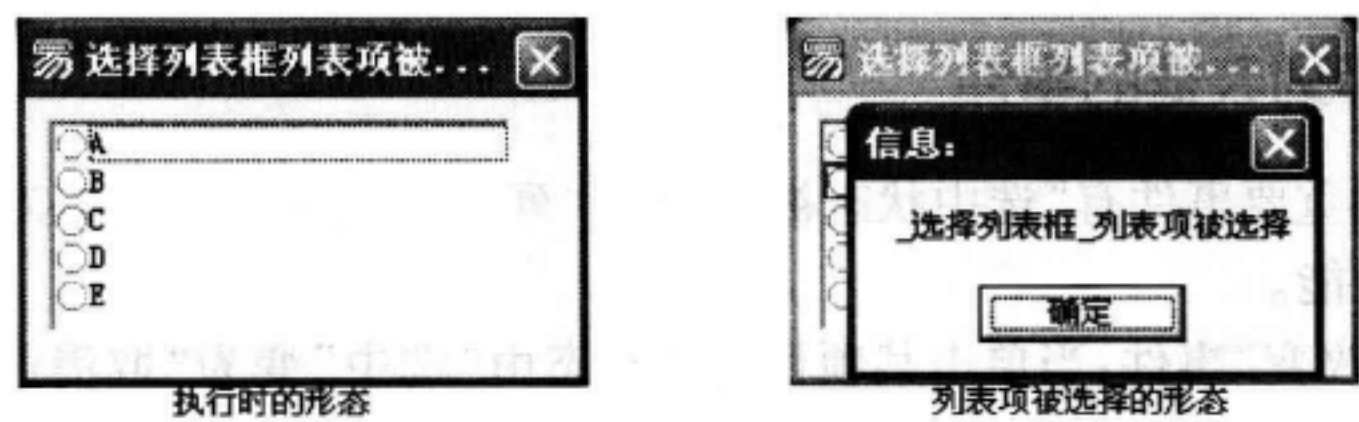


图 8-105 选择列表框“列表项被选择”示例

(2) “双击选择”事件,当使用鼠标左键在某列表项目上双击时产生此事件。如图 8-106 所示,具体参考例程 8-76。

信息框(“_选择列表框_双击选择”,0,)

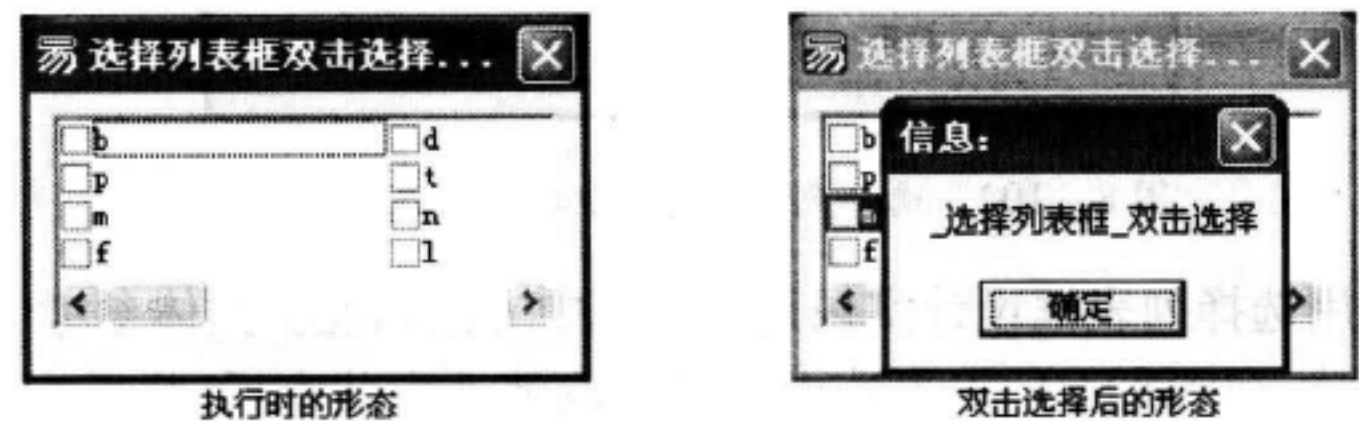



图 8-106 选择列表框“双击选择”示例

第 9 章 进度与时间组件

9.1 进度条组件

9.1.1 进度条概述

进度条组件  是易语言系统提供的显示进度的工具之一,它通过颜色条在框中的步进速度以及数字百分比显示程序执行的进度。只能用代码设置其位置及变化,不可以手工拖动位置。进度条指示操作的进度情况,通常用在耗时较长的操作,如拷贝文件或多重循环中。

进度条没有自己的方法,下面重点介绍进度条的属性和事件。

9.1.2 进度条的属性

进度条的重要属性有“位置”、“最小位置”、“最大位置”、“方向”、“显示方式”等。

“位置”属性描述进度条的当前位置。内容为整数型,它的值应该总在“最小位置”(0)和“最大位置”(100)之间。在程序中可以读取和设置该属性,赋值时,一般不设为较小数,否则无法正确显示,通常可以使用“取整()”或“绝对取整()”命令得到整数,如图 9-1 所示。

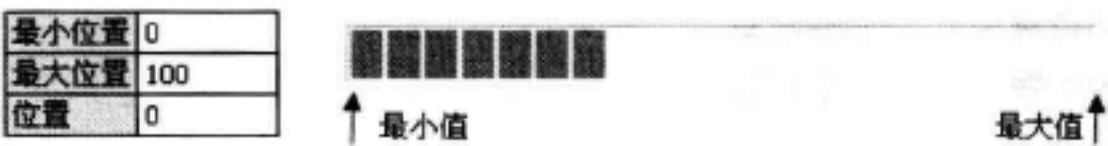


图 9-1 进度条“位置”

“方向”属性控制进度条的横向或纵向。内容为整数型,有两个可选值:“0. 横向”、“1. 纵向”,默认为“0. 横向”。通常横向的比较多一些,纵向的也有。如图 9-2 所示,具体参考例程 9-1。

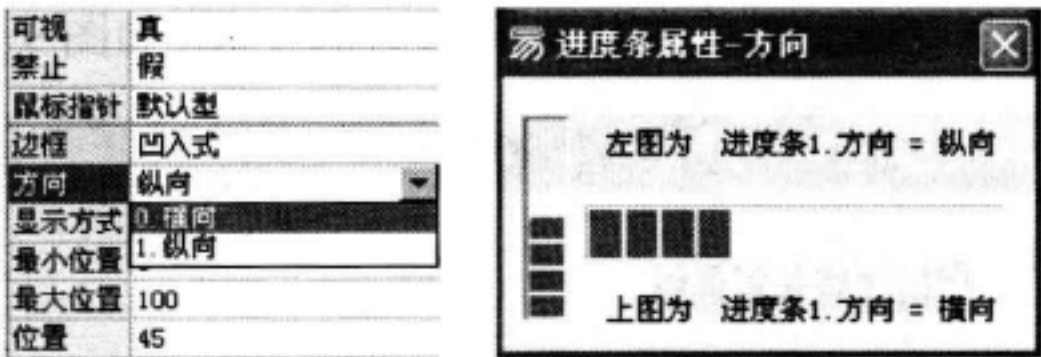


图 9-2 进度条“方向”

“显示方式”属性负责控制进度条是分块显示或连续显示。内容为整数型,有两个可选值:“0. 分块”、“1. 连续”,默认为“0. 分块显示”。如图 9-3 所示,具体参考例程 9-2。

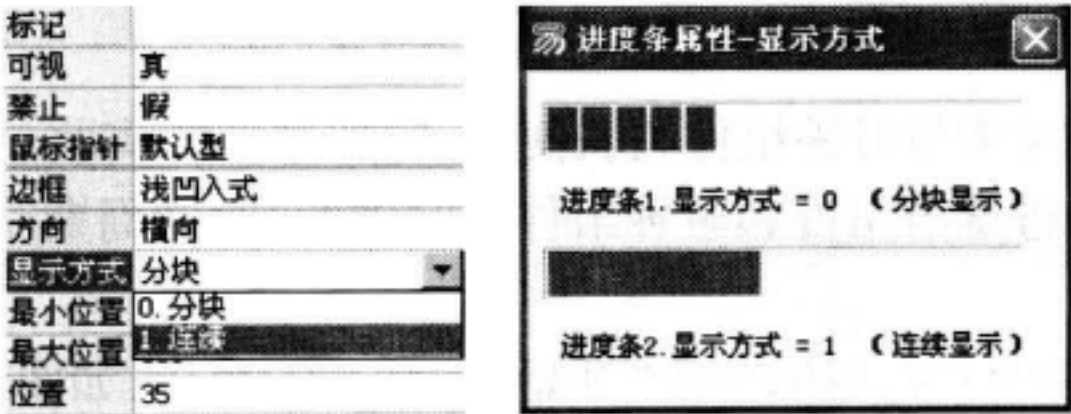


图 9-3 进度条“显示方式”

9.1.3 进度条的事件

进度条的重要事件有“鼠标左键被按下”、“鼠标左键被放开”、“鼠标右键被按下”、“鼠标右键被放开”、“被双击”、“鼠标位置被移动”。基本上,进度条的事件使用得不多,但在个别情况下还是有一定作用的。下面通过简单的例子认识其功能。

【范例 9-1】利用 6 个选择框来验证进度条的事件。具体参考例程 9-3。

下面用一个易语言程序来演示进度条的事件,首先新建一个易程序,在窗口中添加 6 个选择框组件和一个进度条组件,进度条组件的鼠标左键被按下事件和鼠标左键被放开事件被触发的代码如图 9-4 所示。

窗口程序集名	保留	保留	备注
窗口程序集1			

子程序名	返回值类型	公开	备注		
_进度条1_鼠标左键被放开	逻辑型				
参数名	类型	参考	可空	数组	备注
横向位置	整数型				
纵向位置	整数型				
功能键状态	整数型				

还原 0
选择框2.选中 = 真

子程序名	返回值类型	公开	备注		
_进度条1_被双击	逻辑型				
参数名	类型	参考	可空	数组	备注
横向位置	整数型				
纵向位置	整数型				
功能键状态	整数型				

还原 0
选择框3.选中 = 真

图 9-4 进度条事件代码示例

【运行结果】运行例程 9-3,观测单击单选框前后的变化,如图 9-5 所示。

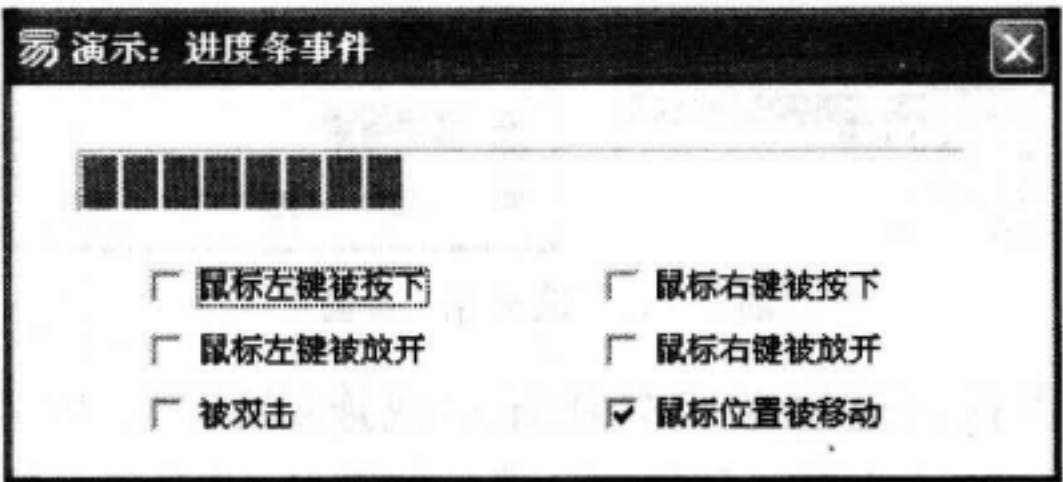


图 9-5 进度条事件运行结果示例

【范例 9-2】进度条通常与时钟组件关联使用,实现进度的演示。具体参考例程 9-4。

例程中有两个组件:进度条组件和时钟组件。时钟的“时钟周期”属性已设置为 100,如图 9-6 所示。

【运行结果】运行例程 9-4,观测单击“开始”按钮后的变化,如图 9-7 所示。

【注意】基本上,进度条的事件使用得不多,但在个别情况下还是有一定作用的。

窗口程序集名	保留	保留	备注
窗口程序集1			

子程序名	返回值类型	公开	备注
_按钮1_被单击			

时钟1.时钟周期 = 200

子程序名	返回值类型	公开	备注
_时钟1_周期事件			

进度条1.位置 = 进度条1.位置 + 1
透明标签1.标题 = 到文本 (进度条1.位置) + “%”
如果 (进度条1.位置 >= 100)
 时钟1.时钟周期 = 0

图 9-6 进度条事件代码示例

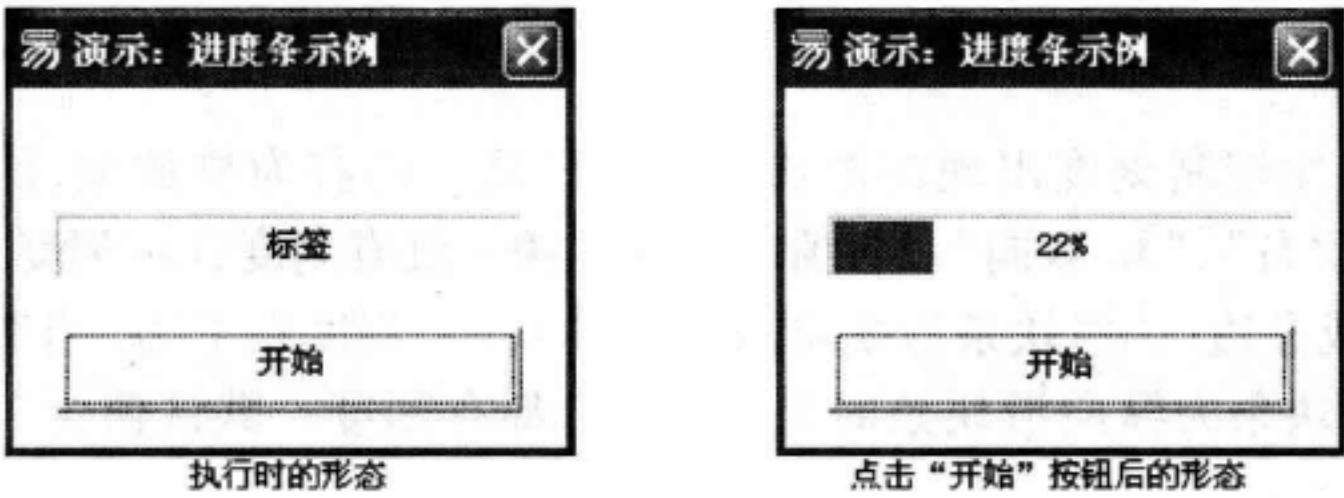



图 9-7 进度条事件运行结果示例

9.2 滑块条组件

9.2.1 滑块条概述

滑块条组件  和进度条组件相类似,它可以手工拖动位置,上面有刻度和滑块用来确定位置。滑块条常用来控制程序的进程等量的属性,比如直接调整音量的大小。滑块条没有自己的方法,下面重点介绍滑块条的属性和事件。

9.2.2 滑块条的属性

滑块条的主要属性有“位置”、“最小位置”、“最大位置”、“行改变值”、“页改变值”、“方向”、“刻度类型”、“单位刻度值”、“允许选择”、“首选择位置”、“选择长度”等。“位置”属性描述滑块条的当前位置,与进度条的同名属性含义类似,内容为整数型。“最小位置”、“最大位置”属性分别描述进度条的最小、最大位置,内容均为整数型,默认值分别为 1 和 10,如图 9-8 所示。



图 9-8 滑块条“位置”

“行改变值”属性指定当用户拖动滑块(或滑块条拥有焦点时按箭头键)时滑块条位置每次增减的数值,内容为整数型,默认为 1。

“页改变值”属性指定当用户在滑块条的空隙处单击或滑块条拥有焦点时按翻页键 Page-Up、PageDown 时滑块条位置增减的数值,内容为整数型,默认为 5。

“方向”属性控制滑块条是横向的还是纵向的。内容为整数型,有两个可选值:“0. 横向”、“1. 纵向”,默认为“0. 横向”。如图 9-9 所示,具体参考例程 9-5。

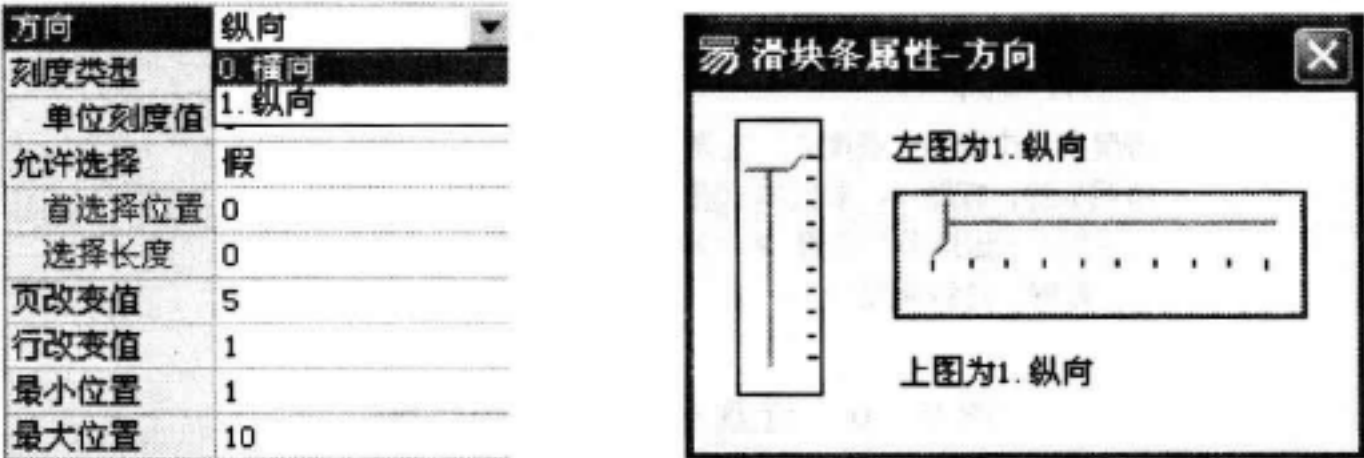


图 9-9 滑块条“方向”

“刻度类型”属性控制刻度出现在滑块条中的位置。内容为整数型,值可以为“0. 无”、“1. 上/左”、“2. 下/右”、“3. 双向”。各值的含义是:0 - 没有刻度;1 - 刻度在上边(当滑块条为横向滑块条时)或左边(当滑块条为纵向滑块条时);2 - 刻度在下边(当滑块条为横向滑块条时)或右边(当滑块条为纵向滑块条时);3 - 两边都有刻度。默认值为“2. 下/右”。如图 9-10 所示,具体参考例程 9-6。



图 9-10 滑块条“刻度类型”

“单位刻度值”属性设置相邻刻度之间的位置差。内容为整数型,默认为 1。

“允许选择”、“首选位置”、“选择长度”属性默认情况下,“允许选择”为“假”,均不生效(首选位置、选择长度只有在允许选择 = 真时才有效)。如图 9-11 是设置允许选择 = 真、首选位置 = 15、选择长度 = 25 后的滑块条外观。具体参考例程 9-7。

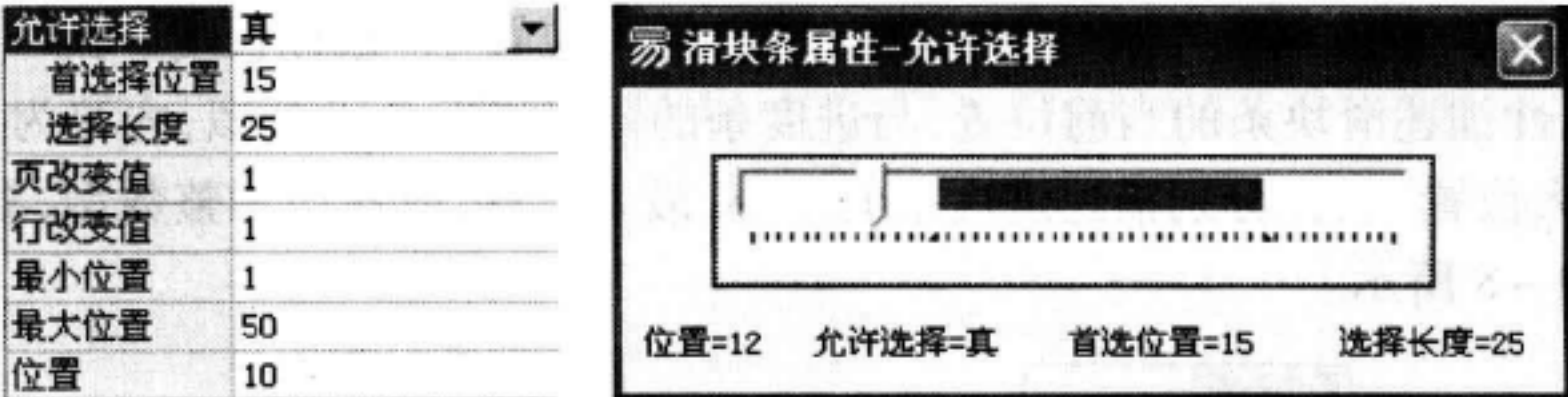


图 9-11 滑块条“允许选择”

9.2.3 滑块条的事件

滑块条的主要事件有“位置被改变”。当滑块的位置被改变(如用户拖动滑块,或在滑块条拥有输入焦点时按箭头键或翻页键)时产生此事件。用代码改动滑块条的位置属性不会引

发该事件。通常在此事件的处理子程序中,读取滑块条的位置属性,并做相应操作。下面通过简单的例子认识其功能。

【范例9-3】用滑块条设计实现位置被改变事件功能。具体参考例程9-8。
通过调节红色、绿色和蓝色中的滑块条来改变颜色。具体的程序代码如图9-12所示。

窗口程序集名	保留	保留	备注
窗口程序集1			

子程序名	返回值类型	公开	备注
_滑块条1_位置被改变			

标签1.背景颜色 = 取颜色值 (滑块条1.位置, 滑块条2.位置, 滑块条3.位置)
' 取颜色值 (r, g, b) ——根据红、绿、蓝分量获取颜色, 各参数均在0-255之间。

子程序名	返回值类型	公开	备注
_滑块条2_位置被改变			

_滑块条1_位置被改变 () ' 直接调用事件处理子程序

' 上一行也可写作: 标签1.背景颜色 = 取颜色值 (滑块条1.位置, 滑块条2.位置, 滑块条3.位置)
' 既然跟 “_滑块条1_位置被改变” 子程序中的代码完全一致, 直接调用该子程序就是了。

子程序名	返回值类型	公开	备注
_滑块条3_位置被改变			

_滑块条1_位置被改变 ()

图9-12 滑块条“位置被改变”代码示例

【运行结果】运行例程9-8,观测移动红色、绿色和蓝色滑块条后的变化,如图9-13所示。

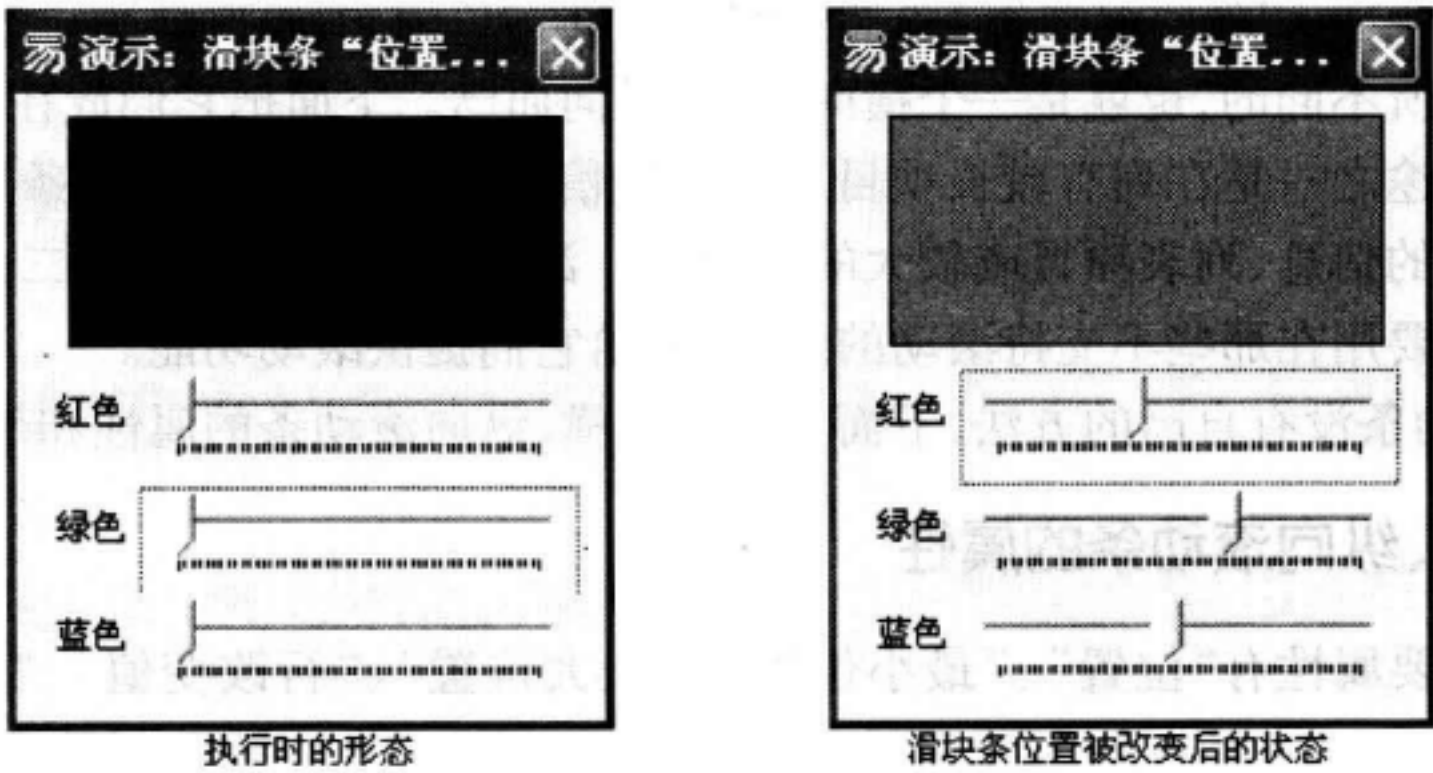


图9-13 滑块条“位置被改变”运行结果示例

【注意】通常在此事件的处理子程序中,读取滑块条的位置属性,并做相应操作。
【范例9-4】用滑块条设计实现播放器功能。具体参考例程9-9。
设计一个简单的播放器实时播放进度,包括两项功能:播放 mp3 音乐和用滑块条显示播放进度。具体的程序代码如图9-14所示。
【运行结果】运行例程9-9,观测单击“开始”后的变化,如图9-15所示。
【注意】通过一个滑块条、一个“选择 mp3”按钮、一个“开始”按钮、一个时钟控件来实现播放 mp3 并显示播放进度的功能。本例程的难点在于显示播放进度。

窗口程序集名	保留	保留	备注
窗口程序集1			
变量名	类型	数组	备注
拖放对象	拖放对象		
文件号	整数型		

子程序名	返回值类型	公开	备注
_打开按钮_被单击			

通用对话框1. 打开 0
mp3 = 通用对话框1. 文件名

子程序名	返回值类型	公开	备注
_时钟1_周期事件			

滑块条1. 位置 = 滑块条1. 位置 + 1

子程序名	返回值类型	公开	备注
_播放按钮_被单击			

播放MP3 (1, mp3)
文件号 = 打开文件 (mp3, 1, 1)
滑块条1. 位置 = 0
滑块条1. 最大位置 = 取文件长度 (文件号) ÷ 16007
时钟1. 时钟周期 = 1000
关闭文件 (文件号)

图 9-14 滑块条事件代码示例

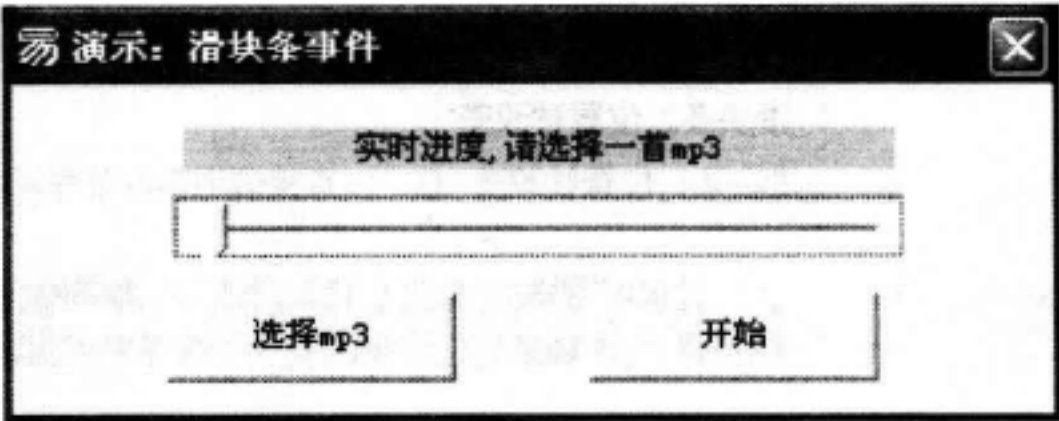


图 9-15 滑块条事件运行结果示例

9.3 横、纵向滚动条组件

9.3.1 横、纵向滚动条概述

横向滚动条和纵向滚动条虽然是两个组件,但它们的属性都是一致的,使用方法也相同。它们之间所不同的,也就是一个横向、一个纵向而已。下面把它们放在一起介绍。

滚动条的用途之一是在列有较长项目或者大量信息的地方,通过滚动条用户可以在小区域中查看到较多的信息、列表项目或较大的图形等。滚动条组件的用途之二是作为数值调整的微调工具。主要用在那些不支持滚动的组件中,给它们提供滚动功能。

横、纵向滚动条没有自己的方法,下面重点介绍横、纵向滚动条的属性和事件。

9.3.2 横、纵向滚动条的属性

滚动条的主要属性有“位置”、“最小位置”、“最大位置”、“行改变值”、“页改变值”、“允许拖动跟踪”。

“位置”、“最小位置”、“最大位置”各属性含义与进度条、滑块条的同名属性基本一致。滚动条“位置”属性的描述如图 9-16 所示。

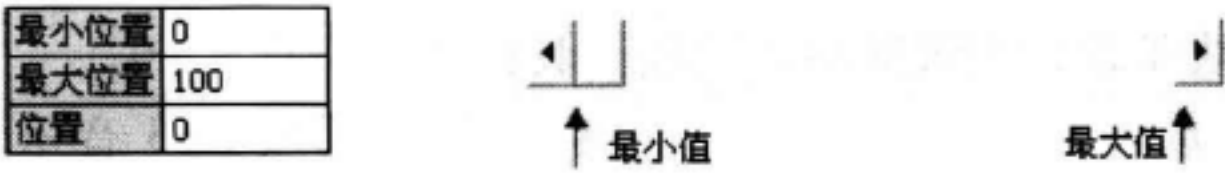


图 9-16 滚动条“位置”

滚动条已分为横向滚动条、纵向滚动条两个组件,所以没有方向属性。

“行改变值”属性指定当用户拖动滑块或滚动条拥有焦点时按箭头键时,滚动条位置每次

增减的数值,内容为整数型,默认为1。

“页改变值”属性指定当用户在滚动条的空隙处单击或滚动条拥有焦点时按翻页键 Page-Up、PageDown 时,滚动条位置增减的数值,内容为整数型,默认为10。

“允许拖动跟踪”属性控制用户拖动滚动条的滑块过程中是否产生“位置被改变”事件。内容为逻辑型,只能为真或假,默认为真。如果本属性为假,则用户拖动滚动条的滑块“过程中”不会产生“位置被改变”事件,只在拖动“结束后”才产生一次“位置被改变”事件。

9.3.3 横、纵向滚动条的事件

滚动条的主要事件有“位置被改变”。下面通过一个简单的例子认识其功能。

【范例9-5】用滚动条设计实现播放器功能。具体参考例程9-10。

设计一个简单的播放器实时播放进度,包括两项功能:播放 mp3 音乐和用滚动条显示播放进度。具体的程序代码如图9-17所示。

窗口程序集名	保留	保留	备注
窗口程序集1			
变量名	类型	数组	备注
拖动对象	拖动对象		
文件号	整数型		

子程序名	返回值类型	公开	备注
_打开按钮_被单击			

通用对话框1. 打开 ()
mp3 = 通用对话框1. 文件名

子程序名	返回值类型	公开	备注
_时钟1_周期事件			

横向滚动条1. 位置 = 横向滚动条1. 位置 + 1

子程序名	返回值类型	公开	备注
_播放按钮_被单击			

播放MP3 (1, mp3)
文件号 = 打开文件 (mp3, 1, 1)
横向滚动条1. 位置 = 0
横向滚动条1. 最大位置 = 取文件长度 (文件号) ÷ 16007
时钟1. 时钟周期 = 1000
关闭文件 (文件号)

图9-17 滚动条“位置被改变”代码示例

【运行结果】运行例程9-10,观测左右移动横向滚动条后的变化,如图9-18所示。

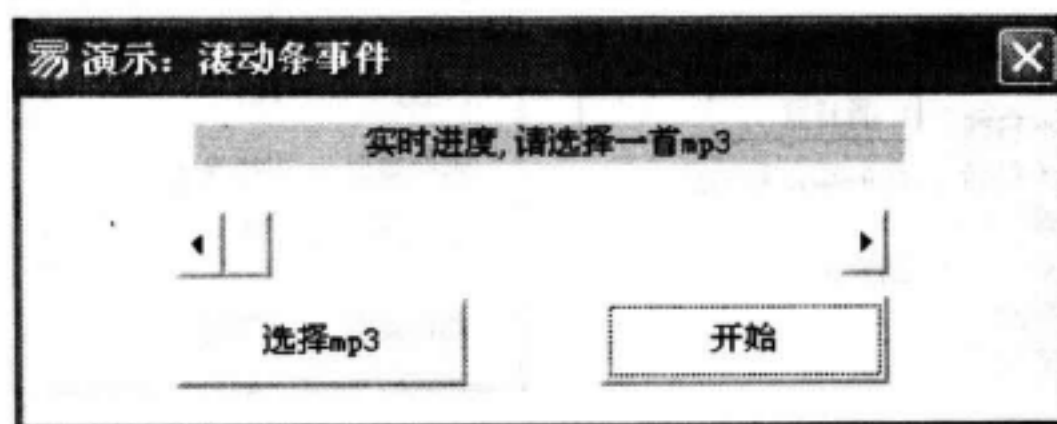


图9-18 滚动条事件运行结果示例

【注意】通过一个滚动条、一个“选择 mp3”按钮、一个“开始”按钮、一个时钟控件来实现播放 mp3 并显示播放进度的功能。本例程的难点在于显示播放进度。

9.4 日期框组件

9.4.1 日期框概述

日期框组件外观上跟颜色选择器、组合框类似,有一个下拉框,只有用户单击时才弹出,如图9-19所示,所以它占用的窗口空间很小。具体参考例程9-11。

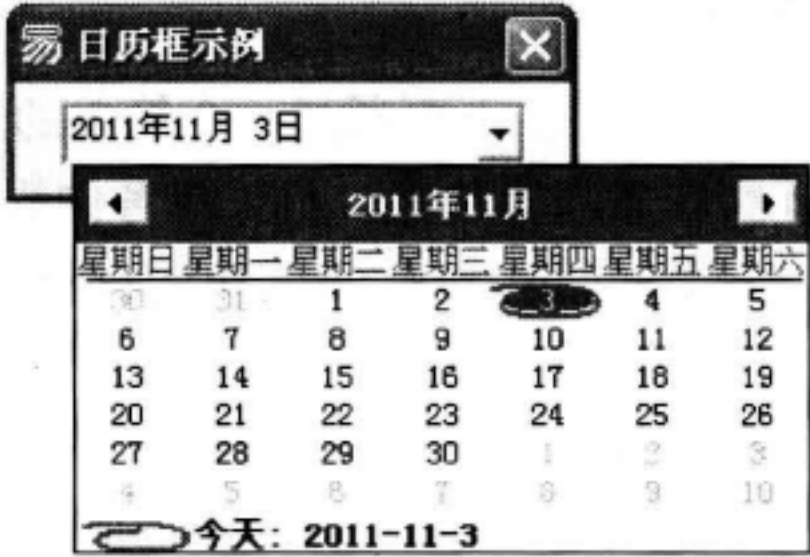


图9-19 日期框示例

日期框没有自己的方法,下面重点介绍日期框的属性和事件。

9.4.2 日期框的属性

日期框的重要属性有“今天”、“附件类型”、“允许编辑”等。

“允许编辑”属性内容为逻辑型。当其值为“假”时,只可以对数字进行编辑;当其值为“真”时,可以编辑日期框内的全部内容。如图9-20所示,具体参考例程9-12。



图9-20 日期框“允许编辑”

“附件类型”属性有两个选择:0. 下拉月历、1. 调节器。当为下拉月历时,会弹出月历面板以供选择,如果为调节器时,只能用调节器改变数字。如图9-21所示,具体参考例程9-13。

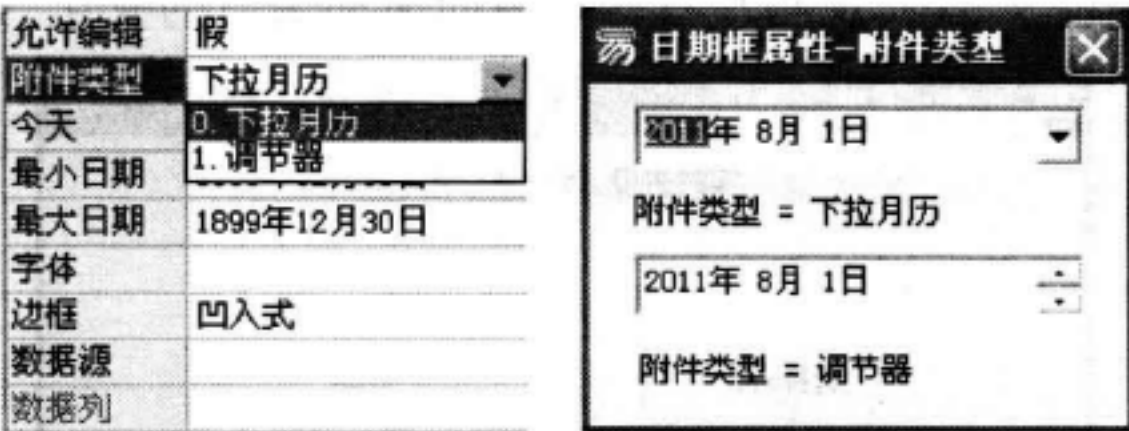


图9-21 日期框“允许编辑”

“今天”属性指出日期框中圈定的日期。本属性的数据类型是日期时间型。通常从“今天”属性中读取用户选择的日期。如图9-22所示,具体参考例程9-14。

编辑框 1. 内容 = 时间到文本(日期框 1. 今天,#日期部分)



图 9-22 日期框“允许编辑”

9.4.3 日期框的事件

日期框的重要事件有“选择日期被改变”,当用户选择了日期框中的某个日期后即产生此事件。下面通过一个简单的例子认识其功能。

【范例 9-6】用日期框设计实现选择日期被改变事件功能。具体参考例程 9-15。从日期框控件中选择一个日期后,该日期就会被写到编辑框中。如图 9-23 所示。

窗口程序集名	保留	保留	备注
窗口程序集1			

子程序名	返回值类型	公开	备注
_日期框1_选择日期被改变			

编辑框1.内容 = 时间到文本 (日期框1.今天, #日期部分)

- “今天”属性是日期框(和月历)的最重要的属性,它指定了用户选择的日期。
- “今天”的数据库类型是“日期时间型”,而编辑框的“内容”属性是“文本型”,所以需要进行数据类型转换。

图 9-23 “选择日期被改变”代码示例

【运行结果】运行例程 9-15,观测选择日期后的变化,如图 9-24 所示。

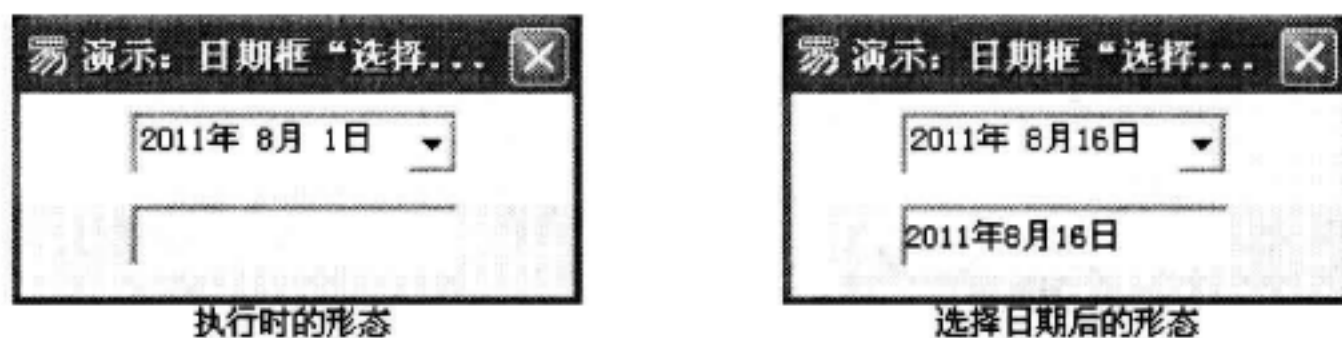



图 9-24 “选择日期被改变”运行结果示例

【注意】当选择不同的日期,就会产生“选择日期被改变”事件,并且编辑框中会显示被选择的日期。

9.5 月历组件

9.5.1 月历概述

月历组件  虽然占用的空间较大,但其属性较多,可以有更多的控制。月历是易语言系统提供的一个具有实用性查找日期的工具,它以列表的形式显示当前设置的日期。用户可以在月历中直接用鼠标修改当前的日期。利用月历,可以查找×年×月的当前日期是星期几。

月历没有自己的方法,下面重点介绍月历的属性和事件。

9.5.2 月历的属性

月历的重要属性有“今天”、“附件类型”、“允许编辑”等。

“不显示今天”属性内容为逻辑型。默认值为“假”，即显示今天。如图 9-25 所示，具体参考例程 9-16。



图 9-25 月历“不显示今天”

“不圈注今天”属性内容为逻辑型。默认值为“假”，即圈注今天。如图 9-26 所示，具体参考例程 9-17。



图 9-26 月历“不圈注今天”

“开始星期首日”属性表示第一竖排显示的是星期几。如图 9-27 所示，具体参考例程 9-18。



图 9-27 月历“开始星期首日”

“显示星期序号”属性内容为逻辑型。该属性的值为“真”时，在左侧第一竖排将显示当前月份在一年当中是第几周。如图 9-28 所示，具体参考例程 9-19。



图 9-28 月历“显示星期序号”

“滚动月数”属性表示单击最上排的左右两个按钮时,前进或后退几个月份,默认为“0”,表示滚动一个月,如设置2、3……分别表示一次将前进或后退2、3……个月。

“今天”属性同日期框的“今天”属性,用于指出月历中圈定的日期。本属性的数据类型是日期时间型。通常从今天属性中读取用户选择的日期。

编辑框1. 内容 = 时间到文本(月历1. 今天,#日期部分)

“允许选择多天”属性内容为逻辑型。如果要选择多天,那么可以使本属性为“真”,如果只选择一天的话,请保持本属性为“假”。当本属性为“真”时,下面一排的“最多选择天数”属性由灰变亮,可以在里面填入数字,此数字即表示可以同时选择的天数最大值。如图9-29所示,具体参考例程9-20。

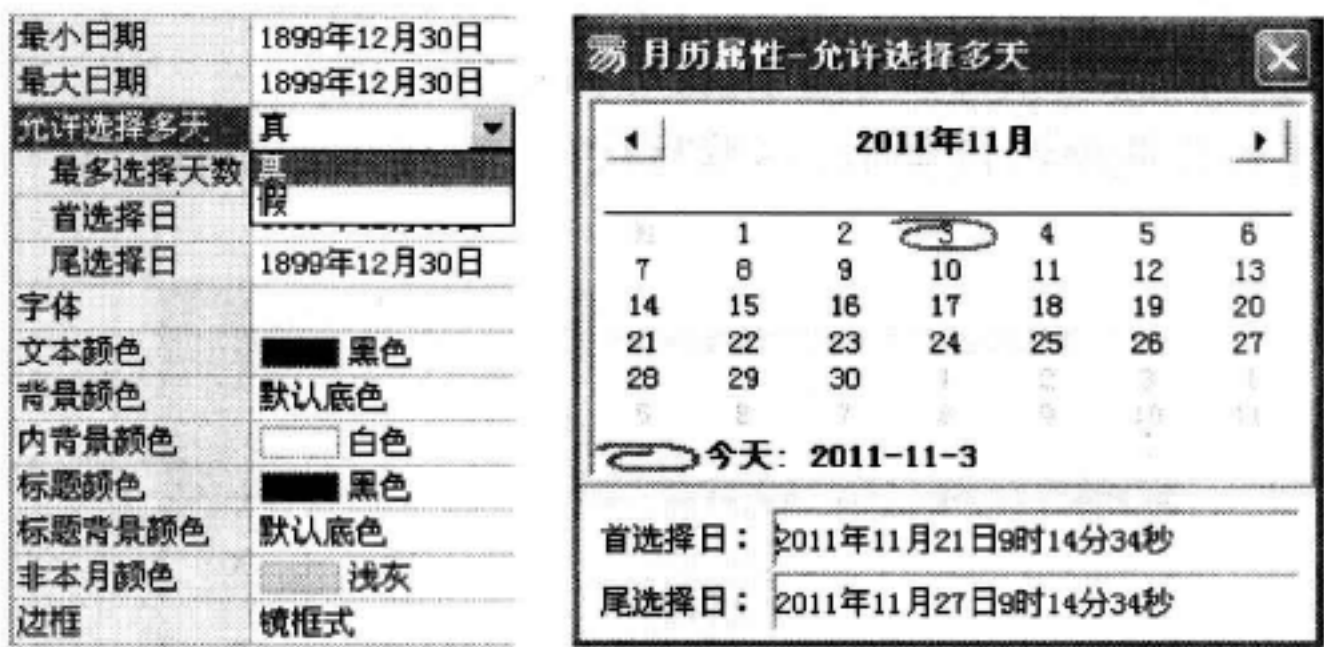


图9-29 月历“允许选择多天”

9.5.3 月历的事件

月历的重要事件有“选择日期被改变”。当现行被选择的日期或区域被改变后产生此事件。下面通过一个简单的例子认识其功能。

【范例9-7】用月历设计实现选择日期被改变事件功能。具体参考例程9-21。从月历控件中选择一个日期后,该日期就会被写到编辑框中。如图9-30所示。

窗口程序集名	保留	保留	备注
窗口程序集1			

子程序名	返回值类型	公开	备注
_月历1_选择日期被改变			

编辑框1. 内容 = 到文本 (月历1. 首选择日)

图9-30 “选择日期被改变”代码示例

【运行结果】运行例程9-21,观测选择日期后的变化,如图9-31所示。




图9-31 “选择日期被改变”运行结果示例

【注意】当选择不同的日期,就会产生“选择日期被改变”事件,并且编辑框中会显示被选择的日期及具体时刻。

9.6 农历日期框组件

9.6.1 农历日期框概述

农历日期框组件是易语言推出的新组件,用于显示和操作农历日期。这就为习惯使用农历的用户和需要编写和农历有关软件的用户提供了方便,并且农历日期支持库中还提供了很多方便的命令,如“农历转公历()”、“公历转农历()”、“取农历节气()”等,可以很容易地进行农历和公历之间的转换和取得关于中国农历的相关信息。

农历日期框组件和日期框组件类似,不编写代码,直接添加组件后运行便可以显示农历,如图 9-32 所示。



图 9-32 展示了“易 农历日期框示例”窗口的界面。窗口顶部显示了当前选择的农历日期为“辛卯(兔)-2011 十月”，对应的公历日期为“辛卯年(兔)十月初八 2011-11-3”。下方是一个 7 列的日历表格，列头为星期一至星期日。表格内容如下：

星期一	星期二	星期三	星期四	星期五	星期六	星期日
廿八 24	廿九 25	三十 26	十月 27	初二 28	初三 29	初四 30
初五 31	初六 11.1	初七 2	初八 3	初九 4	初十 5	十一 6
十二 7	十三 8	十四 9	十五 10	十六 11	十七 12	十八 13
十九 14	廿十 15	廿一 16	廿二 17	廿三 18	廿四 19	廿五 20
廿六 21	廿七 22	廿八 23	廿九 24	冬月 25	初二 26	初三 27
初四 28	初五 29	初六 30	初七 12.1	初八 2	初九 3	初十 4

窗口底部显示了当前的农历节气：“十三立冬 廿八小雪”。

图 9-32 农历日期框示例

下面重点介绍农历日期框的属性、方法和事件。

9.6.2 农历日期框的属性

“公历日期”属性内容为日期时间型,用于设置当前日期的公历日期。该属性改变后,“农历年”、“农历月”、“农历日”属性也会随之改变,改变成和当前公历日期相对应的农历日期。

“农历年”、“农历月”、“农历日”属性内容都为整数型,3 个属性用来设置完整的当前日期的农历日期。当这 3 个属性任意一个改变后,“公历日期”属性也会随之改变成与其相对应的公历日期。

“是否闰月”属性设置当前月是否是农历闰月。如农历 1998 年 5 月有闰月,所以农历除了有五月还多了闰五月(即 1998 年闰年农历有 13 个月),如果设置农历日期框组件的“农历月”属性为 5 月,当“是否闰月”属性为假,则当前农历月份设置的为五月,如果“是否闰月”属性为真,则当前农历月份为闰五月。该属性不是取得当前月是否为闰月的属性,而是设置当前月是否是闰月。

“主视图”属性有 2 个选项:0. 农历、1. 公历,用于设置农历日期框组件以公历为主显示还是以农历为主显示。如图 9-33 所示。



图 9-33 农历日期框“主视图”

“农历年份”、“农历月份”、“农历日份”这三个属性同为文本型只读属性,“农历年份”属性则返回当前年的干支年号(干支年即为甲子、乙丑、丙寅……),“农历月份”和“农历日份”属性则返回当前月、日的汉字大写文本。如农历日期框组件的当前农历日期为 2011 年 10 月 8 日,则上述三个属性分别为“辛卯”、“十”、“八”。

“属相”属性内容为文本型只读属性,返回当前年的属相。“节气”属性内容为文本型只读属性,如果当日为农历的节气,则该属性返回节气的名称;返回空文本则表示当天不是节气。“星期”属性内容为整数型,只读属性,属性值 1 表示星期一,2 表示星期二,……,7 表示星期日。

“农历本月天数”、“农历本年天数”属性内容为整数型,只读属性,分别返回农历本月的天数和本年的天数。

“格式化文本”属性内容为文本型,用于描述要显示的农历日期格式。其中可以包含以下格式化字符:% Y、% S、% M、% D、% y、% m、% d,其各自代表不同的意义。% Y 表示以干支表示的农历年份,如“甲辰”;% S 表示属相;% M 表示农历月;% D 表示农历日;% y 表示阳历年份;% m 表示阳历月份;% d 表示阳历日。如果本属性为空,则默认显示格式为“% Y 年(% S)% M% D % y - % m - % d”。该属性默认的情况下,显示的文本为“辛 卯年(兔)十月初八 2011-11-3”。点击农历日期框属性“格式化文本”的按钮,使用以下代码可将显示的文本重定义。如图 9-34 所示。

农历日期框 1. 格式化文本 = “今年为:% Y 年,今年属相为:<% S>,今天:% M% D”



图 9-34 农历日期框“格式化文本”

“显示文本”属性内容为文本型只读属性,用于设置显示于窗口组件中的文本。该文本的内容是依据“格式化文本”属性格式化后的日期信息。

前面讲到的农历日期框的几个属性示例,具体参考例程 9-22。

9.6.3 农历日期框的方法

农历日期框的方法有“置农历日期()”、“置公历日期()”、“增减日期()”、“打开选择窗口()”和“关闭选择窗口()”。

(1) “置农历日期()”方法负责设置当前农历日期。设置当前农历日期也可以通过对“农历年”、“农历月”和“农历日”属性赋值,但使用该方法设置当前日期更为简便。

调用格式:

〈逻辑型〉农历日期框. 置农历日期(农历年,农历月,农历日,[是否闰月])

参数说明:

“农历年”、“农历月”、“农历日”都为整数型,分别填写欲设置为当前农历日期的年、月、日。

农历日期框 1. 置农历日期(2010,11,1,)

(2) “置公历日期()”方法负责设置当前的公历日期。

调用格式:

〈逻辑型〉农历日期框. 置公历日期(公历年,公历月,公历日)

参数说明:

“公历年”、“公历月”、“公历日”都为整数型,分别填写欲设置为当前公历日期的年、月、日。

农历日期框 1. 置公历日期(2010,11,1,)

(3) “增减日期()”方法负责在当前日期的基础上增减指定天数。

(4) “打开选择窗口()”、“关闭选择窗口()”方法分别负责在代码中实现打开和关闭日期选择窗口。

9.6.4 农历日期框的事件

农历日期框的事件有“日期被选择”、“日期被悬停”、“选择窗口打开”、“选择窗口关闭”和“日期被改变”。下面通过简单的例子认识其功能。

(1) “日期被选择”事件是当用户单击了日期选择窗口中的某个日期时产生的事件。可在事件处理子程序中读取组件的相关属性,以获取用户选择的日期。当使用代码改变当前日期时,不会触发本事件。

置现行时间(农历日期框 1. 2010,11,1,)

(2) “日期被悬停”事件是当用户用鼠标指向日期选择窗口中的某个日期时产生的事件。

(3) “选择窗口打开”事件即当选择窗口打开时触发。

(4) “选择窗口关闭”事件即当选择窗口关闭时触发。

(5) “日期被改变”事件是当日期改变时(用户选择了日期/通过属性设置日期/通过方法设置日期)产生的事件。可在事件处理子程序中读取组件的相关属性,以获取被改变后的日期。当用鼠标选择和改变农历日期框的当前日期时,会先触发“日期被改变”事件,然后触发“日期被选择”事件。但当使用代码改变农历日期框的当前日期时,将只会触发“日期被改变”事件,而不会触发“日期被选择”事件。

【范例9-8】用农历日期框设计实现农历公历间的转化功能。具体参考例程9-23。

在本范例中选择好农历公历之间的转化顺序,再选择好日期并换算即可得到预期结果。具体的程序代码如图9-35所示。

子程序名	返回值类型	公开	备注
换算_被单击			

变量名	类型	静态	数组	备注
日期	日期时间型			
年	整数型			
月	整数型			
日	整数型			

```

转换判断 0
年 = 到数值 (源年数.内容)
月 = 到数值 (源月数.内容)
日 = 到数值 (源日数.内容)
日期 = 指定时间 (年, 月, 日, , , )
如果真 (日期 < [1881年1月30日] 或 年 > 2060)
    目的显示.内容 = ""
    千支显示.内容 = ""
    返回 0
如果 (X = 真)
    年数 = 年
    月数 = 月
    日数 = 日
    启动窗口.农历显示 0
    目的显示.内容 = 到文本 (农历年份) + "年" + 到文本 (启动窗口.农历.标题)
    千支显示.内容 = 到文本 (启动窗口.千支.标题)
    日期 = 农到公历 (年, 月, 日)
    年数 = 取年份 (日期)
    月数 = 取月份 (日期)
    日数 = 取日 (日期)
    目的显示.内容 = "公元" + 到文本 (年数) + "年" + 到文本 (月数) + "月" + 到文本 (日数) + "日"
    启动窗口.农历显示 0
    千支显示.内容 = 到文本 (启动窗口.千支.标题)
  
```

图9-35 农历日期框范例代码示例

【运行结果】运行例程9-23,观测单击“换算”按钮的变化,如图9-36所示。

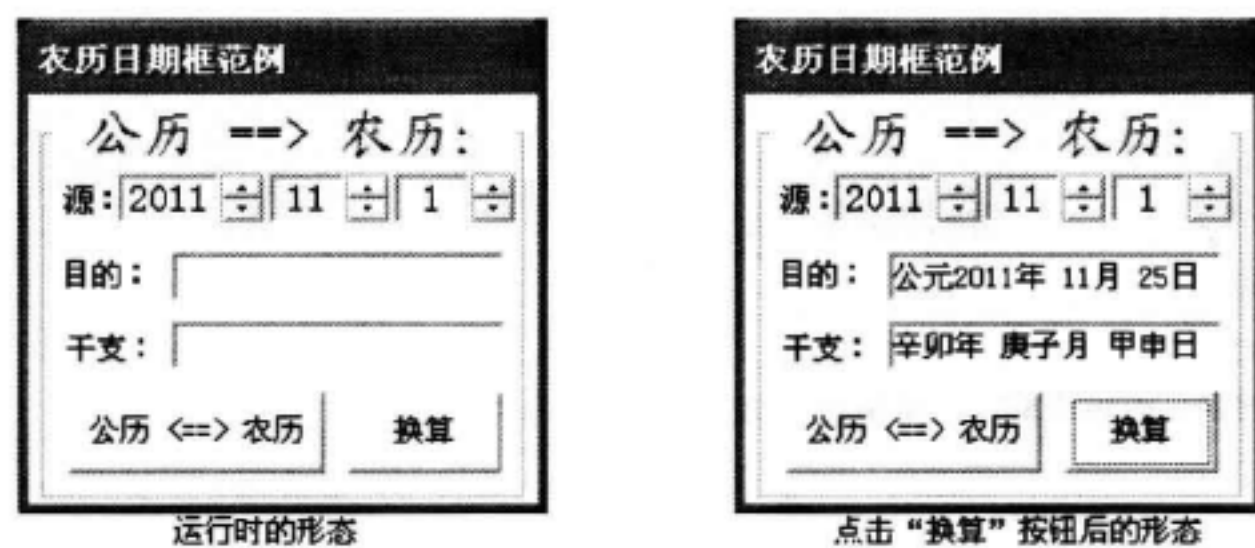


图9-36 农历日期框范例运行结果示例

【注意】农历日期支持库中除了有农历日期框组件外,还包括了农历日期处理类的命令,这些命令极大地方便了对农历日期的相关操作,并且会涉及到一些农历方面的知识,详细的信息大家可以查看相关资料,这里就不再赘述。

9.7 时钟组件

9.7.1 时钟概述

时钟组件^①是一个比较重要的组件,很多计算时间的程序都会用到它。它是最简单又最

常用的组件之一,为非可视组件,即在程序运行时,用户是看不到它的。时钟最大的特点是每隔一定时间,自动产生一个周期事件。

9.7.2 时钟的属性

时钟的主要属性有“时钟周期”属性,负责控制相邻两“周期事件”的间隔时间。内容为整数型,单位是毫秒。只有当时钟周期属性大于0,时钟组件才会自动产生“周期事件”。时钟周期不能为负值,时钟周期属性的值越小,“周期事件”就产生的越频繁。如果该属性等于0,则时钟不再产生“周期事件”(即所说的“停止时钟”)。

9.7.3 时钟的事件

时钟的主要事件有“周期事件”事件。每隔一定时间(由时钟周期指定)自动产生此事件。当时钟周期 >0 时,在程序运行后,会自动执行本事件的处理子程序中的代码。如果时钟周期 $=0$,就意味着“停止时钟”,即停止产生“周期事件”。当此后再次为时钟周期赋 >0 的值时,时钟又会重新“启动”。下面通过简单的例子认识其功能。

【范例9-9】用时钟设计实现周期事件功能。具体参考例程9-24。

此例程运行时,在“启动时钟”前打上勾,并且将滑块条向左边移动一些。可以看到随着周期数的变小,圆色块闪动的速度越来越快,如果将滑块条向右边移动,那么圆色块闪动的速度就会变慢。具体的程序代码如图9-37所示。

窗口程序集名	保留	保留	备注
窗口程序集1			

子程序名	返回值类型	公开	备注
_时钟1_周期事件			本程序每1秒(1000毫秒)被自动调用一次。

```
如果 (外形框1.填充颜色 = #红色)
外形框1.填充颜色 = #黄色
外形框1.填充颜色 = #蓝色
```

子程序名	返回值类型	公开	备注
_选择框1_被单击			

```
如果真 (选择框1.选中 = 真)
    时钟1.时钟周期 = 滑块条1.位置

如果真 (选择框1.选中 = 假)
    时钟1.时钟周期 = 0
    令“时钟周期”=0 意味着“停止时钟”,此后不再产生“周期事件”,
    直到再次为“时钟周期”赋大于0的值。
```

子程序名	返回值类型	公开	备注
_滑块条1_位置被改变			

```
时钟1.时钟周期 = 滑块条1.位置
标签1.标题 = 到文本 (时钟1.时钟周期)
    标签的“标题”属性是文本型的,而时钟的“时钟周期”属性是整数型的,
    要进行数据类型转换后,才能相互赋值。
选择框1.选中 = 真
```

图9-37 “周期事件”代码示例

【运行结果】运行例程 9-24,观测运行后的变化,如图 9-38 所示。

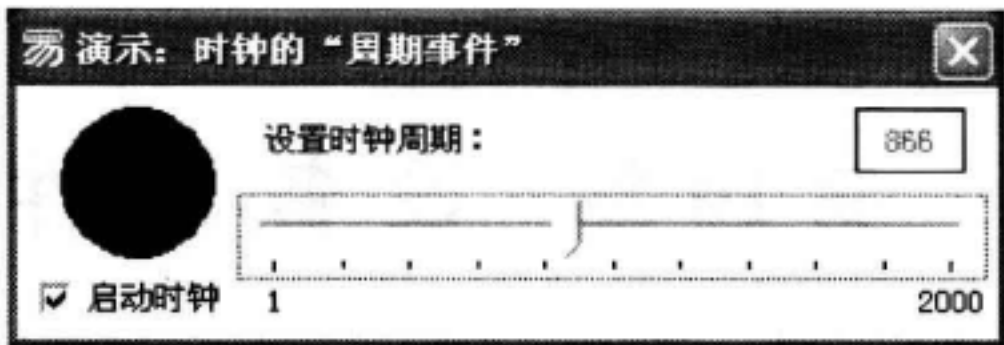


图 9-38 “周期事件”运行结果示例

【注意】“周期事件”是必须响应的,时钟周期属性也是必须设置的。时钟周期不要太小,否则会过多地占用系统资源,“周期事件”中的代码执行时间不能太长,否则可能引起程序崩溃。

【范例 9-10】用时钟设计实现计时器功能。具体参考例程 9-25。

范例中标签组件放在启动窗口的上排,时钟组件放在下排,设置时钟组件的时钟周期属性为 960。具体的程序代码如图 9-39 所示。

窗口程序集名	保留	保留	备注
窗口程序集1			

子程序名	返回值类型	公开	备注
_时钟1_周期事件			

标签1.标题 = 取文本右边 (“0” + 到文本 (取小时 (取现行时间 ()), 2) + “:” + 取文本右边 (“0” + 到文本 (取分钟 (取现行时间 ()), 2) + “:” + 取文本右边 (“0” + 到文本 (取秒 (取现行时间 ()), 2)

图 9-39 “周期事件”代码示例

【运行结果】运行例程 9-25,观测运行后的变化,如图 9-40 所示。

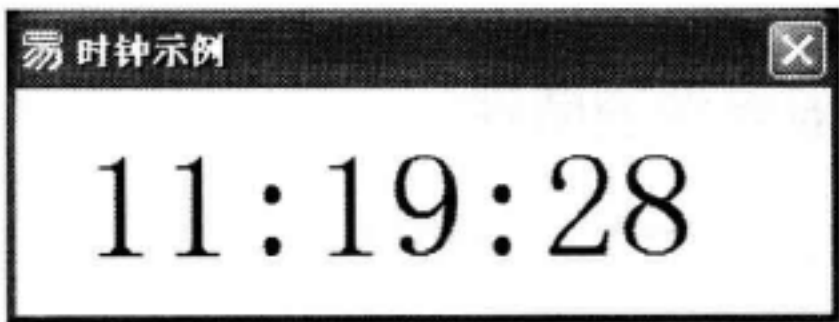


图 9-40 “周期事件”运行结果示例

【注意】这个计时器,使用时钟组件每间隔 960 毫秒即重写一遍标签,而标签样式是时间文本,这样就达到了完成显示时间的目的,看上去就像一个时钟在走。

第 10 章 设备组件

10.1 颜色选择器


颜色选择器组件  在运行时提供一个下拉框,用户可在其中选择任意一种颜色,如图 10-1 所示。



图 10-1 颜色选择器

10.1.1 颜色选择器组件的属性

颜色选择器的重要属性有:颜色、允许透明。

1. “颜色”属性

此属性中保存了颜色选择器中当前被选择的颜色。可在程序运行中把它赋给其他组件的“文本颜色”、“背景颜色”等属性,例如:

标签 1. 文本颜色 = 颜色选择器 1. 颜色

2. “允许透明”属性

此属性控制下拉颜色表中是否出现透明色。该属性值为逻辑型,只能为“真”或“假”,默认为“假”,即不出现透明色。如果本属性为真时,即会在下拉框中第一排显示“透明”字样,如图 10-2 所示。

10.1.2 颜色选择器组件的事件

颜色选择器的重要事件有:颜色被改变。“颜色被改变”事件是当用户从下拉颜色表中选择了一个颜色时产生的。下面通过一个简单的例子认识其功能。

【范例 10-1】在窗口添加颜色选择器组件、标签及编辑框组件(如图 10-3 所示),当选中了某一颜色时,标签的文本颜色变为该颜色,同时编辑框中显示该颜色的数值。具体参考例程 10-1。



图 10-2 颜色选择器“允许透明”属性为真

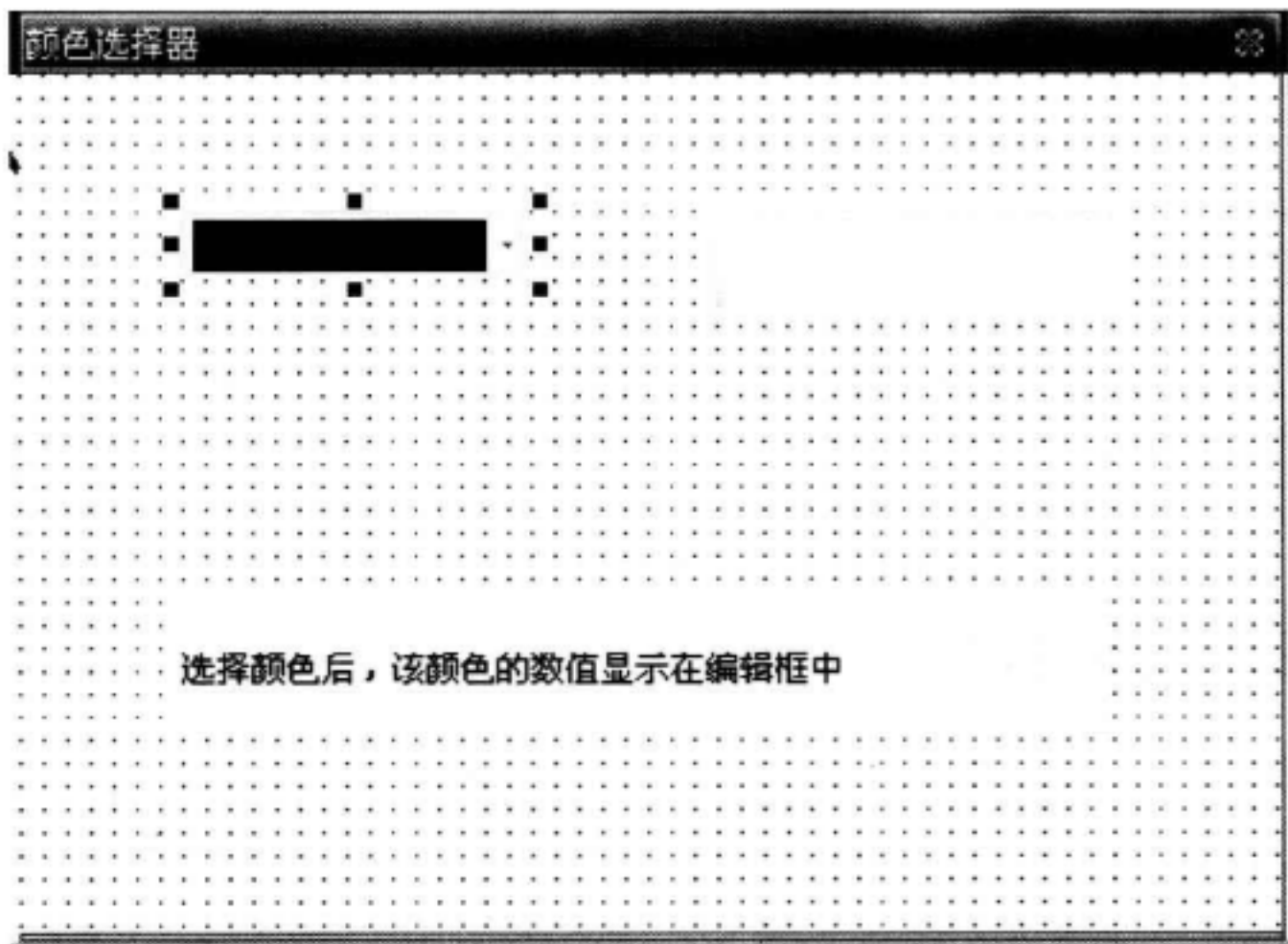


图 10-3 设计窗口

启动窗口的程序代码如图 10-4 所示。

窗口程序集名	保 留	保 留	备 注
窗口程序集1			

子程序名	返回值类型	公开	备 注
_颜色选择器1_颜色被改变			

标签1. 文本颜色 = 颜色选择器1. 颜色

编辑框1. 内容 = 到文本 (颜色选择器1. 颜色)

图 10-4 颜色选择器范例程序集

【运行结果】运行例程 10-1，观测通过颜色选择器选中某一颜色时标签和编辑框的变化，如图 10-5 所示。

【范例解析】当鼠标选择颜色选择器下拉列表中的颜色时，触发“颜色被改变”事件，执行如图 10-3 所示的两条命令，一是标签的标题文本“选择颜色后，该颜色的数值显示在编辑框中”变为此颜色，二是选中颜色的数值显示在编辑框中。

【注意】颜色其实是以整数型值的形式表示和存储。

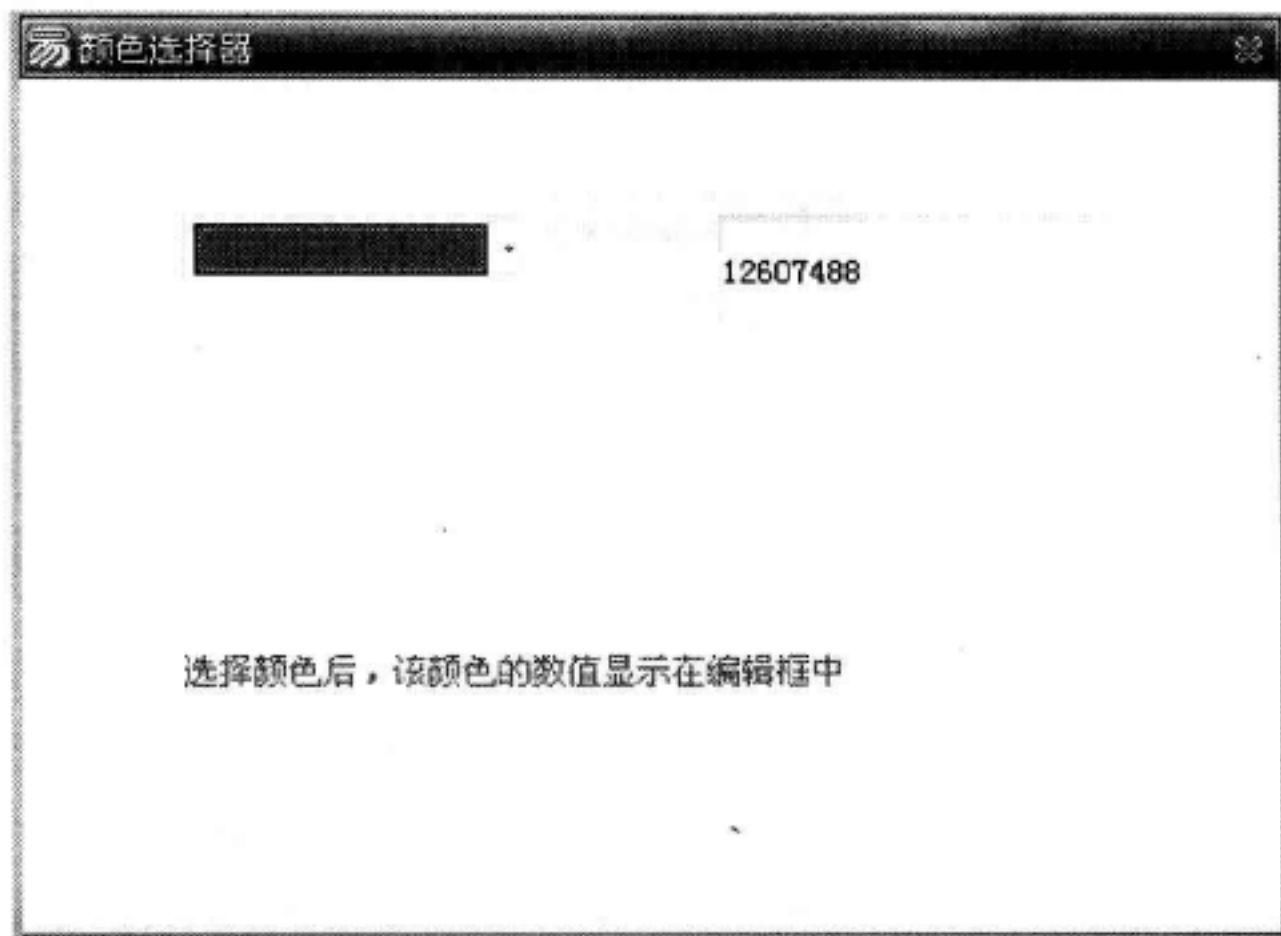



图 10-5 颜色选择器范例运行结果

10.2 打印机组件

打印机组件  是重要的输出组件，是一个非可视组件。使用它，就可以将所要打印的文本或图片打印出来。

10.2.1 打印机组件的属性

打印机的重要属性有：打印份数、多份打印方式、最小页号、最大页号、首页页号、末页页号、打印选择区、左边空、右边空、顶边空、底边空、当前页号、已打印份数、打印区宽度、打印区高度、打印被中止等。

其他属性如画笔颜色、刷子颜色、画笔类型、刷子类型、画出方式、画笔粗细、绘画单位、文本颜色、文本背景颜色、字体等，与画板组件的同名属性含义相同。

打印机组件的属性具体如图 10-6 所示。

1. “左边空”、“右边空”、“顶边空”、“底边空”属性

上述属性可以设置打印的内容出现的位置，例如使打印内容距离纸张的左边和顶边 600 像素，可以用以下代码实现：

```
打印机 1. 左边空 = 600
打印机 1. 顶边空 = 600
```

2. “字体”属性

此属性可以设置打印文本的字体属性，如图 10-7 所示。

在程序中用代码实现如下所示：

```
打印机 1. 字体. 字体名称 = “隶书”
打印机 1. 字体. 字体大小 = 12
打印机 1. 字体. 加粗 = 真
打印机 1. 字体. 倾斜 = 假
打印机 1. 文本颜色 = #蓝色
```


名称	打印机1
备注	
左边	160
顶边	80
宽度	40
高度	16
标记	
绘画单位	0.1毫米
画笔类型	直线
画出方式	复制笔
画笔粗细	0
画笔颜色	黑色
刷子类型	颜色刷子
刷子颜色	白色
文本颜色	黑色
文本背景颜色	透明色
字体	
打印作业名	
打印份数	1
多份打印方式	假
最小页号	-1
最大页号	-1
首页页号	1
末页页号	-1
打印选择区	-1
左边空	0
顶边空	0
右边空	0
底边空	0
当前页号	** 只读属性 **
已打印份数	** 只读属性 **
打印区宽度	** 只读属性 **
打印区高度	** 只读属性 **
打印被中止	** 只读属性 **

图 10-6 打印机组件属性

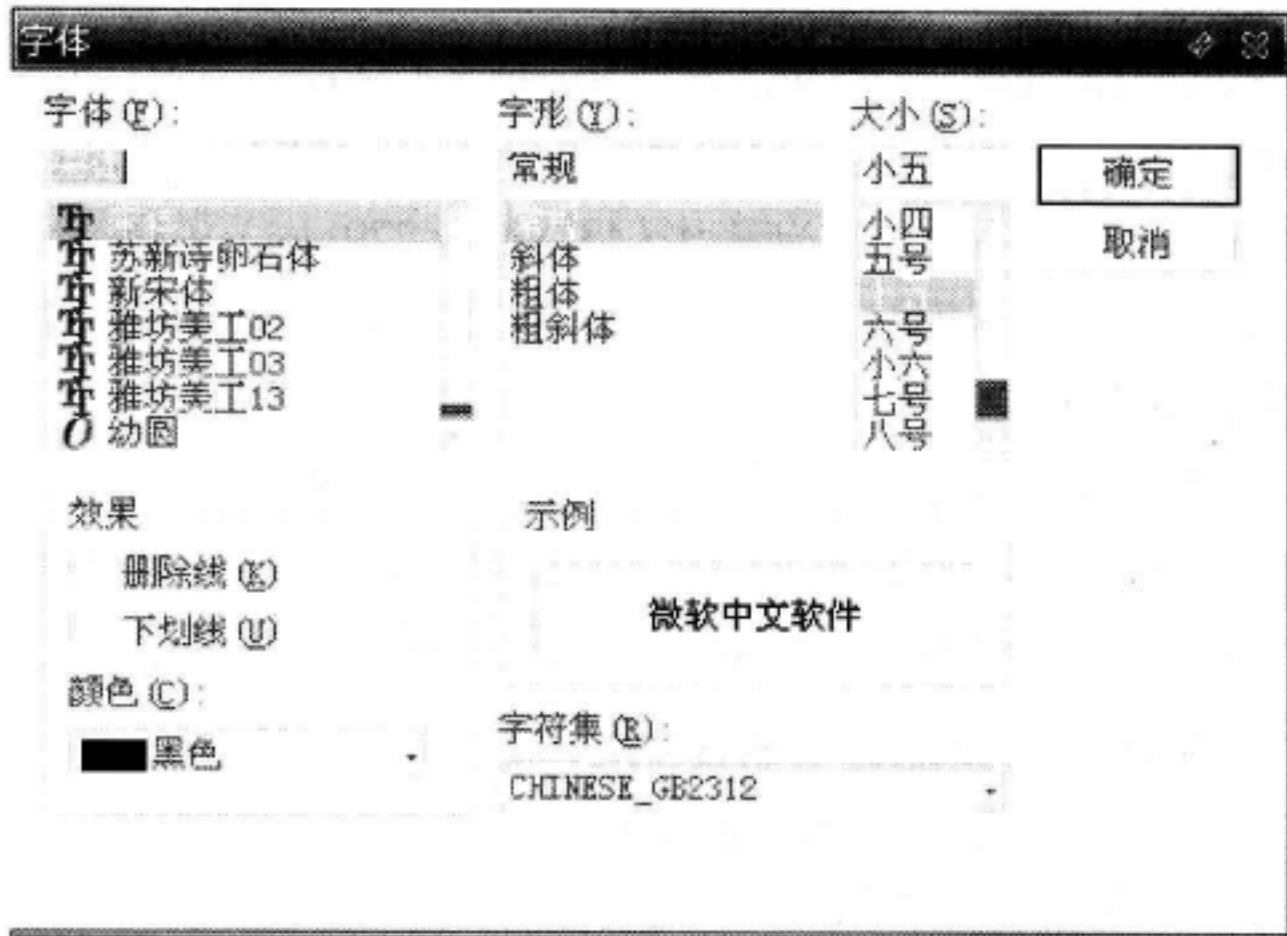


图 10-7 打印机的字体属性

10.2.2 打印机组件的方法

打印机的方法有很多,如图 10-8 所示。其专有方法有:开始打印()、结束打印()、取消打印()、换页()、开始下一份()等。

开始打印()方法的功能是使用户的作业按照设定的方式开始打印,返回“真”表示已经成功地进入了打印作业,进入打印作业后文本写出位置将被重置为 0。如果执行本命令时已经在打印作业中或者用户在打印设置对话框中选择了取消,则返回“假”。进入打印后必须调用“结束打印”或者“取消打印”方法来退出打印。其调用格式如下:

〈逻辑型〉对象. 开始打印([是否调用打印设置对话框], [是否显示打印状态对话框], [纸张], [纸张方向])

(1) 参数“是否调用打印设置对话框”的类型为“逻辑型(bool)”,可以被省略。此参数指定在进入打印之前是否先调用打印设置对话框对打印机及相关打印参数进行设置。如省略本参数,默认为“真”。如果参数值为“假”,将自动使用 Windows 系统默认打印机,但如果取 Windows 系统默认打印机失败,仍然会调用打印设置对话框。

(2) 参数“是否显示打印状态对话框”的类型为“逻辑型(bool)”,可以被省略。此参数指定在打印过程中是否显示由系统自动提供的打印状态对话框。用户如果在该打印状态对



图 10-8 打印机组件的方法

话框中选择了取消打印,系统将自动中止并取消该打印作业,并置“打印被中止”属性值为“真”。如省略本参数,默认为“真”。

(3) 参数“纸张”的类型为“整数型(int)”,可以被省略。此参数指定所使用的打印纸类型。可以为以下常量值之一:(0)#默认纸、(1)#A3 纸(297mm×420mm)、(2)#A4 纸(210mm×297mm)、(3)#A5 纸(148mm×210mm)、(4)#B4 纸(250mm×354mm)、(5)#B5 纸(182mm×257mm)、(6)#4 开(215mm×275mm)、(7)#16 开(146mm×215mm)、(8)#32 开(97mm×151mm)、(9)#信纸(216mm×279mm)、(10)#法律用纸(216mm×355mm)、(11)#行政用纸(184mm×266mm)、(12)#声明(140mm×216mm)、(13)#小报(279mm×432mm)、(14)#笔记(216mm×279mm)、(15)#账本(432mm×279mm)、(16)#对开纸(216mm×330mm)。注意如果所选择纸张得不到打印机的支持,打印机将会自动选择最接近的纸张。如省略本参数,默认为“#默认纸”。

(4) 参数“纸张方向”的类型为“整数型(int)”,可以被省略。此参数指定打印纸的放置方向。可以为以下常量值之一:(0)#纵向、(1)#横向。如省略本参数,默认为“#纵向”。

根据以上解释,可以编写一个最简单的打印程序,代码如下:

```
如果真(打印机 1. 开始打印(,) = 真)
    打印机 1. 写文本行(编辑框 1. 内容)
    打印机 1. 结束打印()
如果真结束
```

这段程序表示将一个编辑框中所有的内容打印出来,打印前没有进行任何的设置。

打印机其他方法如“画图片()、取图片宽度()、取图片高度()、写文本行()、写出()、定位写出()、置写出位置()、画点()、画直线()、画矩形()、画圆角矩形()、画渐变矩形()、画多边形()、画椭圆()、画弧线()、画弦()、画饼()、填充矩形()、取宽度()、取高度()、取设备句柄()”等与画板组件的同名方法用法相同,只不过是一个“画”在画板上、一个“画”在打印纸上而已。如果画板组件用得比较好,使用打印机组件也就轻车熟路了。

【范例 10-2】实现打印机打印图片。设计窗口如图 10-9 所示。运行时在此窗口打开图片,点击“打印图片”按钮时进入图片设置窗口,可以设置打印的图片的尺寸和比例等信息,如图 10-10 所示,设置完成后点击“打印”按钮联机打印输出。具体参考例程 10-2。

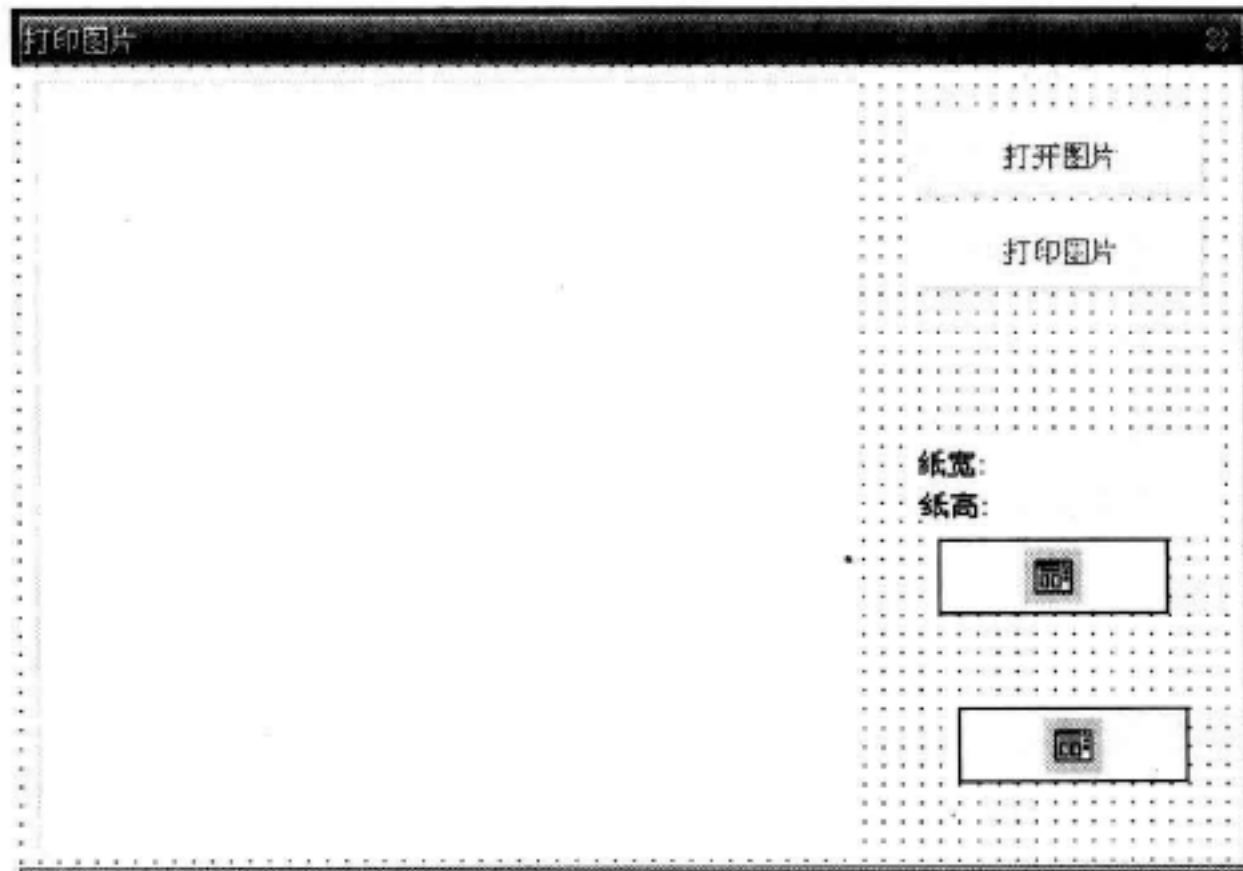


图 10-9 打印图片窗口

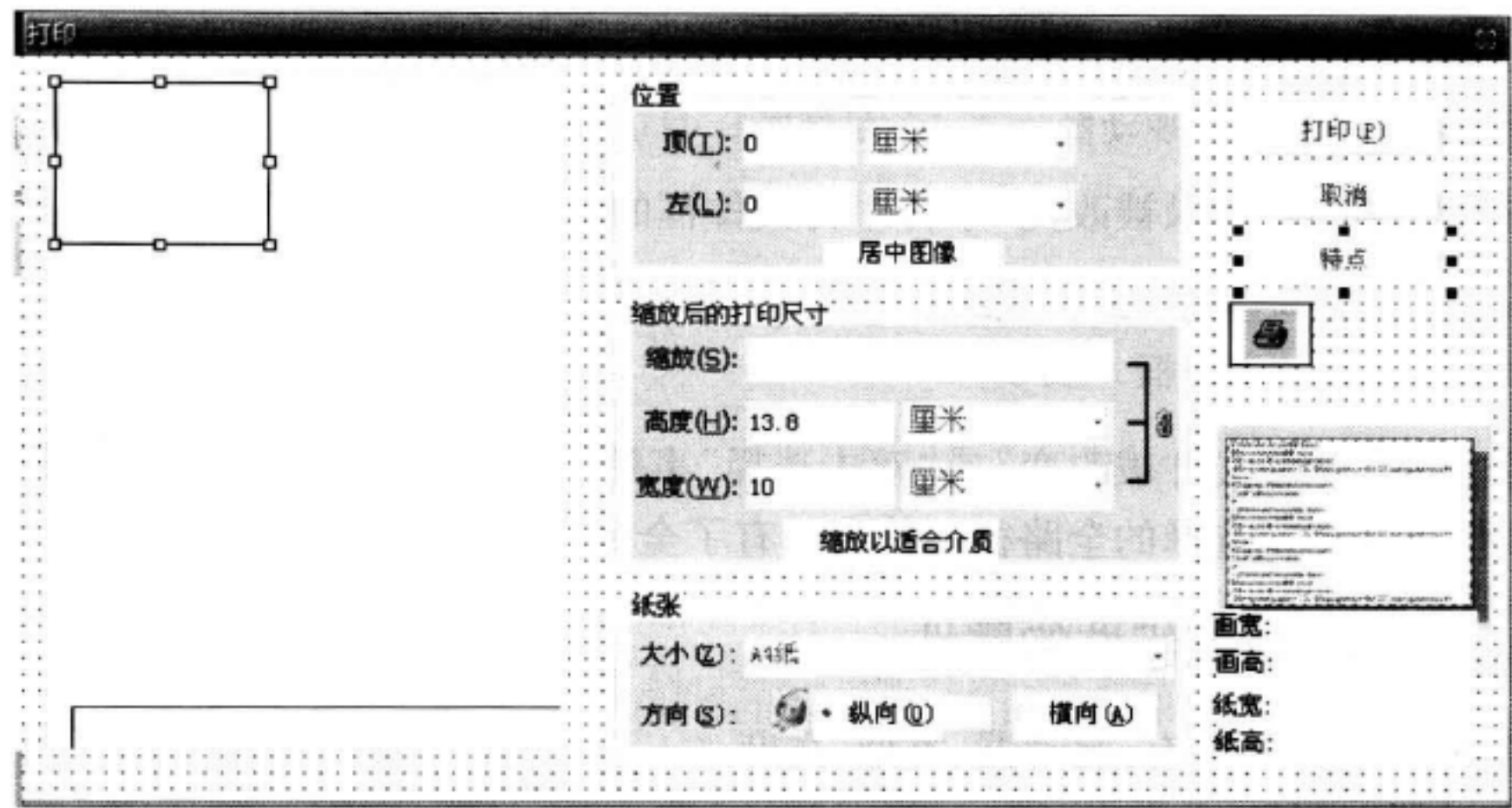


图 10-10 图片设置打印窗口


图片打印的程序代码如图 10-11 所示。

子程序名	返回值类型	公开	备注
_按钮1_被单击			
打印机1.开始打印 (真, 真, ,)			
打印机1.画图片 (图片号, 1, 1, 到数值 (编辑框宽度.内容) × 100, 到数值 (编辑框高度.内容) × 100,)			
打印机1.结束打印 ()			

图 10-11 图片打印程序代码

【范例解析】要打印的图片在打印窗口的左侧编辑框中,可以自由设置高度和宽度,确定后使用打印机的画图片方法将图片根据设置值打印出来。

10.3 驱动器框组件

驱动器组件一般与目录框组件、文件框组件这两个组件配合起来使用,其作用就是共同来定位一个文件,或一个目录。如图 10-12 所示。

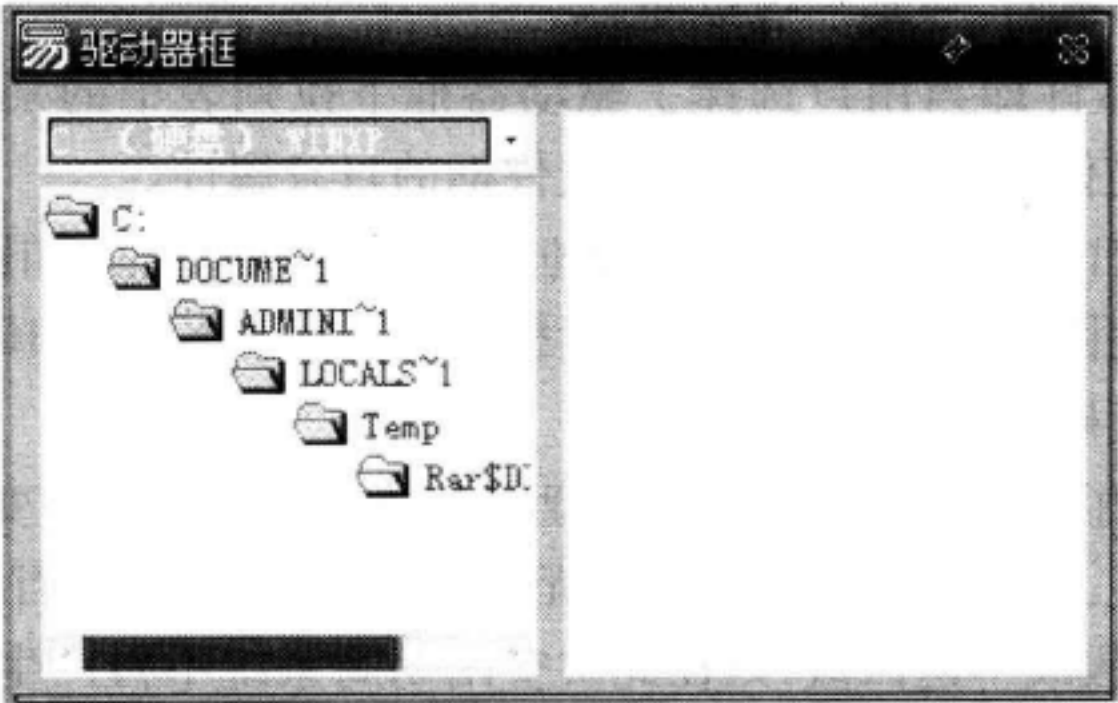


图 10-12 驱动器框、目录框和文件框

驱动器组件的一般应用形式如下：

- (1) 在驱动器框的“驱动器被改变”事件中为目录框、文件框的“目录”属性赋值。常见代

码是:

连续赋值(驱动器框 1. 驱动器 + “:\”, 目录框 1. 目录, 文件框 1. 目录)

(2) 在目录框的“目录被改变”事件中为文件框的“目录”属性赋值。常见代码是:

目录框 1. 目录 = 驱动器框 1. 驱动器 +

文件框 1. 目录 = 目录框 1. 目录

在文件框的“选择文件被改变”或“双击选择”事件中,可以根据文件框的“目录”和“被选择文件”属性得到用户选择的全路径文件名。有了全路径文件名,就可以对文件进行操作了。

10.3.1 驱动器框组件的属性

驱动器框的重要属性有:驱动器、类型、高度。

1. “驱动器”属性

“驱动器”属性为文本型,指当前选择的驱动器,用大写字母 A-Z 表示各盘符。如果在驱动器组件的下拉列表中选择了 C 盘,则驱动器属性的值就是“C”;同样,选择了 D 盘则其值为“D”。本属性的默认值是当前盘的盘符。驱动器属性在设计期是不可改动的。但在程序运行中,可以用代码来读取(或设置)它的值。

读取代码:x = 驱动器框 1. 驱动器

设置代码:驱动器框 1. 驱动器 = “C”

2. “类型”属性

“类型”属性为整数型,指定可用的驱动器类型。共有 6 个可选值,默认值为 0,表示列出全部驱动器。一般保持默认值即可,本属性通常不必设置。

3. “高度”属性

“高度”属性用来限定驱动器框的高度,默认是 20,如果驱动器多的时候只能显示 3 排,其他的驱动器要通过翻页来选择,如果将高度属性设置大一些,可以显示更多的驱动器行数。

10.3.2 驱动器框组件的事件

驱动器框的重要事件有:驱动器被改变。

“驱动器被改变”事件即当用户选择了驱动器框下拉列表中的某个选项时产生的事件。本事件的处理子程序的任务是更新目录框和文件框的目录属性。

【范例 10-3】在窗口添加驱动器框、目录框、文件框及图片框,如图 10-13 所示,运行时选择盘符、路径,选中的图片文件会显示在图片框中。具体参考例程 10-3。

启动窗口的程序代码如图 10-14 所示。

【运行结果】运行例程 10-3,观测选择了盘符和路径后图片框的变化,如图 10-15 所示。

【范例解析】名为“_启动窗口_空闲”的子程序是事件处理程序,当系统处于空闲状态时产生此事件,若返回值为假或不返回值,则在此次空闲期间系统不再产生空闲事件;若返回值为真,则继续产生空闲事件。参数“已空闲时间”指示自系统进入本次空闲状态到现在所经过的时间,单位为毫秒。

在本范例中,当产生了一个空闲事件,就把选中文件的路径和文件名赋值给“现在应显示文件”变量,并将其在图片框中显示出来。

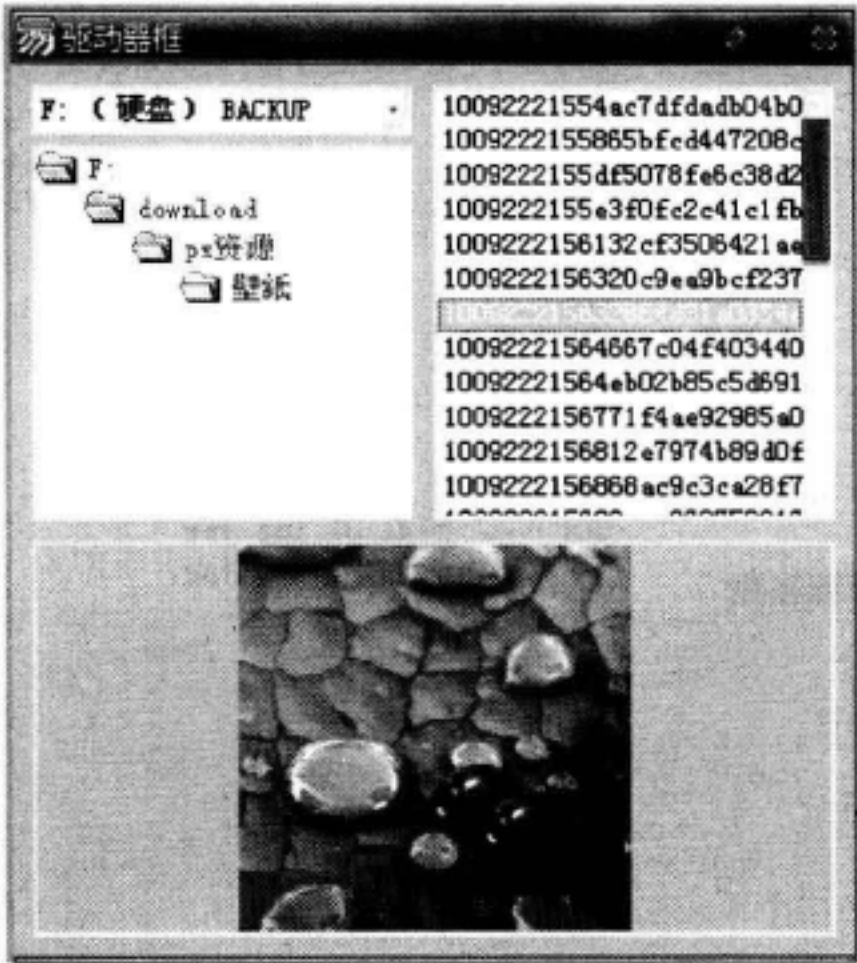


图 10-15 驱动器框范例运行结果

10.4 目录框组件

目录框组件可以显示本机的文件目录,供用户选择查看。上文中已经提及,本组件常与驱动器框组件及文件框组件配合使用。参看图 10-12。

10.4.1 目录框组件的属性

目录框组件的重要属性有:目录。

“目录”属性为文本型,指出目录框中被选中的目录。本属性在设计时不可用。通常在驱动器框的“驱动器被改变”事件中设置目录框的“目录”属性。

10.4.2 目录框组件的事件

目录框组件的重要事件有:目录被改变。

“目录被改变”事件即当用户双击了目录框中的某个文件夹时产生的事件。本事件的处理子程序的任务是更新文件框的“目录”属性。如图 10-16 所示。

子程序名	返回值类型	公开	备注
_目录框1_目录被改变			

文件框1.目录 = 目录框1.目录

图 10-16 目录被改变事件代码

【范例 10-4】启动窗口添加一个“保存”按钮和一个编辑框,在编辑框中输入文字,点击“保存”按钮,进入保存窗口中选择存储地址,将其存储为 txt 文本文件。启动窗口和保存窗口分别如图 10-17 和图 10-18 所示。具体参考例程 10-4。

保存窗口的程序代码如图 10-19 所示。

【范例解析】在此范例中,定义了“_启动窗口.标记”作为一个全局变量使用,在启动窗口和保存窗口之间传递变量。启动窗口中编辑框的文本内容将保存在保存窗口中选定的存储路径中。

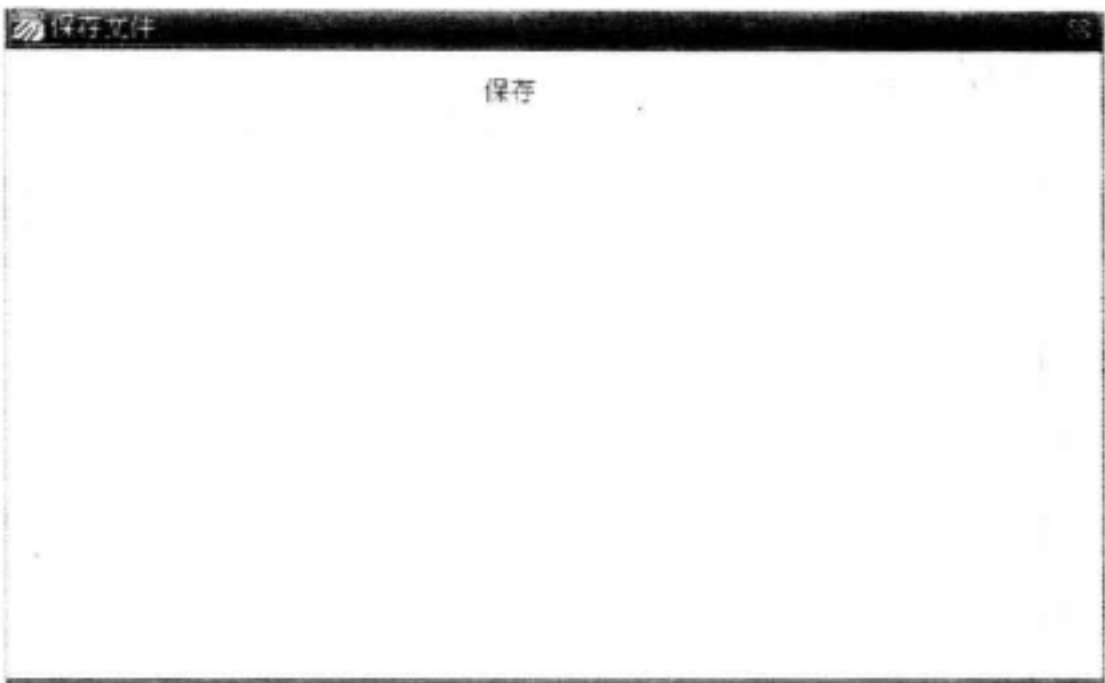


图 10-17 启动窗口

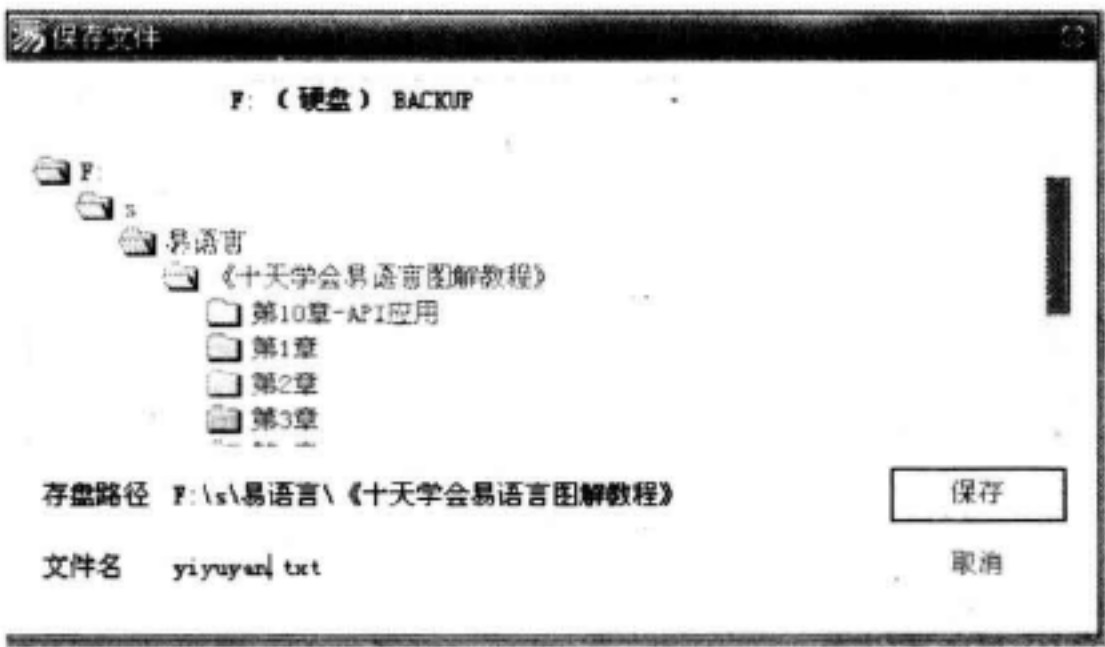


图 10-18 设置文件的存盘路径

窗口程序集名	保留	保留	备注
保存窗口程序集			

子程序名	返回值类型	公开	备注
_保存窗口_创建完毕			

编辑框1.内容 = 目录框1.目录

子程序名	返回值类型	公开	备注
_驱动器框1_驱动器被改变			

连续赋值 (驱动器框1.驱动器 + “\”, 目录框1.目录)

子程序名	返回值类型	公开	备注
_目录框1_目录被改变			

编辑框1.内容 = 目录框1.目录

子程序名	返回值类型	公开	备注
_按钮2_被单击			

销毁 0

子程序名	返回值类型	公开	备注
_按钮1_被单击			


_启动窗口.标记 = 编辑框1.内容 + “\” + 编辑框2.内容

信息框 (_启动窗口.标记, 0,)

销毁 0

图 10-19 保存窗口程序代码

10.5 文件框组件

文件框组件可以显示当前目录下的文件名,供用户选择查看。上文中已经提及,本组件常与驱动器框组件及目录框组件配合使用。请参看图 10-12 所示。

10.5.1 文件框组件的属性

文件框组件的重要属性有：目录、被选择文件、允许选择多项、通配符等。

1. “目录”属性

此属性在设计时不可用，文件框只显示指定目录中的文件。

2. “被选择文件”属性

此属性指出文件框中被选择文件的文件名（有扩展名，但不含路径）；如果有多个文件被选择，各文件名之间用半角分号“；”隔开。

3. “允许选择多项”属性

此属性值为逻辑型，指是否允许同时选择多个文件，默认为“假”，不允许。

4. “通配符”属性

此属性指定允许进入文件框的文件类型。可以同时指定多个通配符，各通配符之间用半角分号“；”隔开。如：“*.exe;*.com;*.bat”，文件框中只显示 EXE、COM、BAT 三种类型的文件，其他的全被过滤掉。

文件框还有一些其他属性，如“通常”、“存档”、“只读”、“系统”、“隐藏”属性，这几个属性都是逻辑型的，与文件框内单个文件的属性有关，如设置只读属性为“真”，那么文件框内文件中，只要有只读的文件才会显示出来，否则不显示。

10.5.2 文件框组件的事件

文件框组件的重要事件有：选择文件被改变、双击选择。

1. “选择文件被改变”事件

当用户选择了文件框中的另一个文件时产生“选择文件被改变”事件。

2. “双击选择”事件

当用户双击了文件框中的某一个文件时产生“双击选择”事件。

这两个事件中，均可以通过代码来获取被选择文件的全路径文件名。

【范例 10-5】启动窗口添加一个“打开”按钮和一个编辑框，点击“打开”按钮，在打开窗口中选择文本文件，其内容会显示在启动窗口的编辑框内。启动窗口和打开窗口分别如图 10-20 和图 10-21 所示。具体参考例程 10-5。

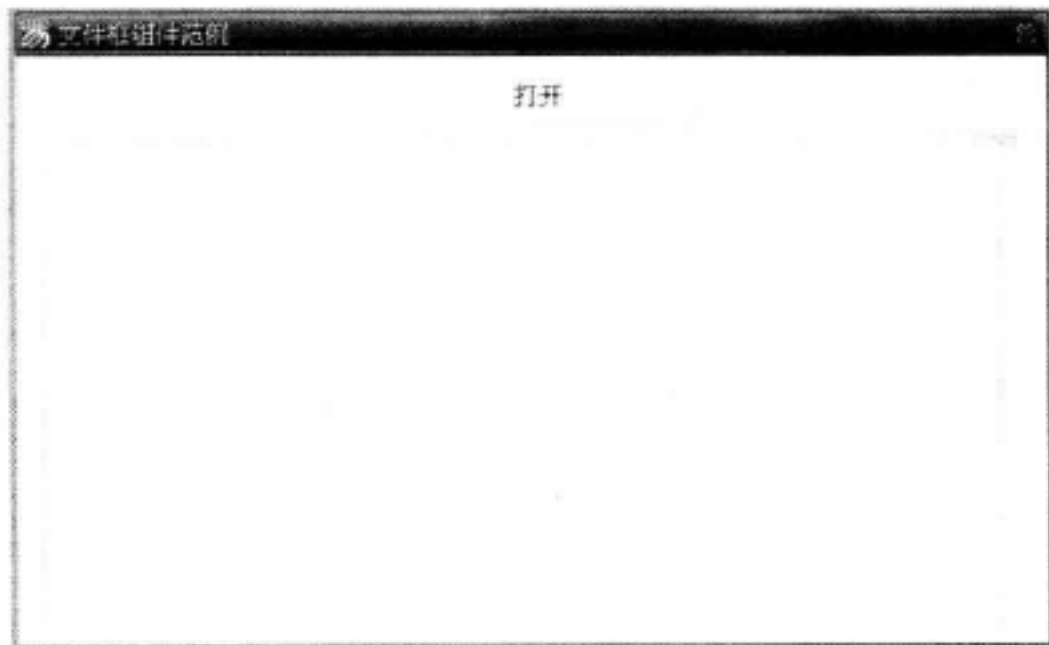


图 10-20 启动窗口

启动窗口的程序代码如图 10-22 所示。

打开窗口的程序代码如图 10-23 所示。

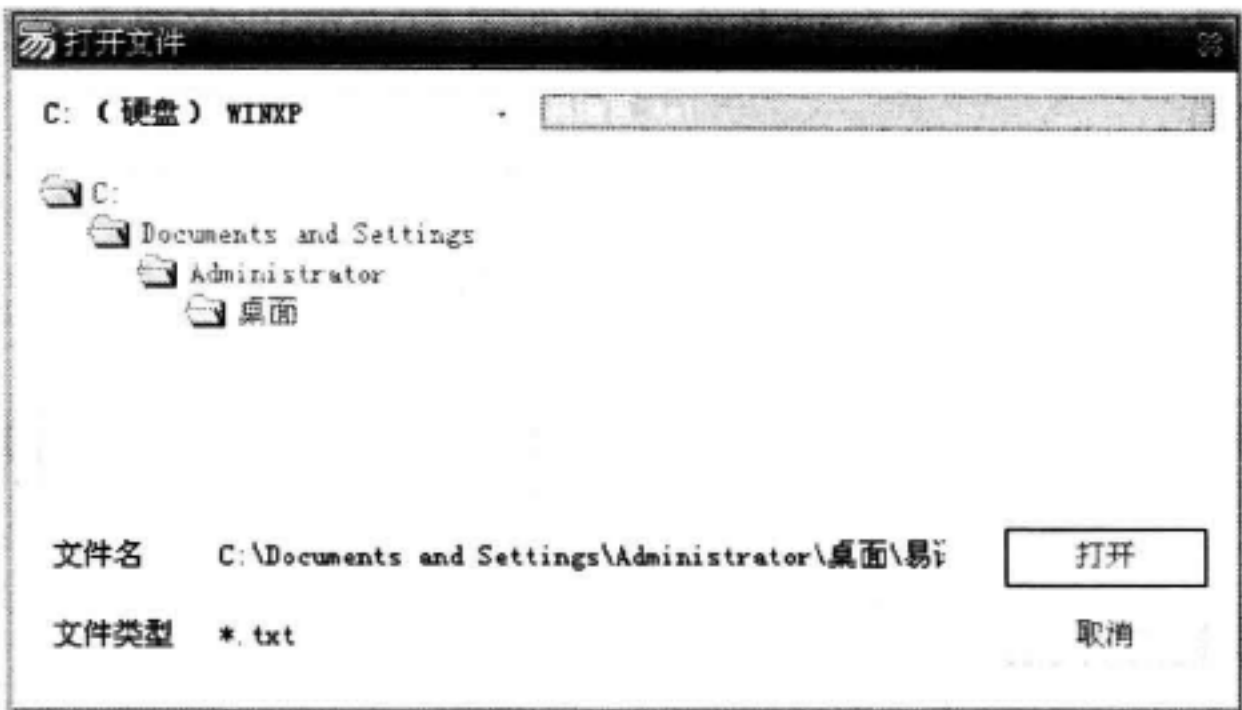


图 10-21 打开窗口

窗口程序集名	保留	保留	备注
窗口程序集1			

子程序名	返回值类型	公开	备注
_按钮1_被单击			

变量名	类型	静态	数组	备注
文件号	整数型			

载入 (打开窗口, , 真)

文件号 = 打开文件 (启动窗口. 标记, #读入,)

如果 (文件号 ≠ 0)

 编辑框1. 内容 = 读入文本 (文件号,)

 关闭文件 (文件号)

信息框 ("打开文件失败!", #错误图标, "错误")

图 10-22 启动窗口程序代码

子程序名	返回值类型	公开	备注
_目录框1_目录被改变			

文件框1. 目录 = 目录框1. 目录

子程序名	返回值类型	公开	备注
_文件框1_选择文件被改变			

编辑框1. 内容 = 目录框1. 目录 + "\" + 文件框1. 被选择文件

子程序名	返回值类型	公开	备注
_按钮2_被单击			

销毁 0

子程序名	返回值类型	公开	备注
_按钮1_被单击			

_启动窗口. 标记 = 编辑框1. 内容

销毁 0

子程序名	返回值类型	公开	备注
_文件框1_双击选择			

编辑框1. 内容 = 目录框1. 目录 + "\" + 文件框1. 被选择文件

_按钮1_被单击 0

图 10-23 打开窗口程序代码

【运行结果】运行例程 10-5, 观测选择了文本文件后编辑框的变化, 如图 10-24 所示。

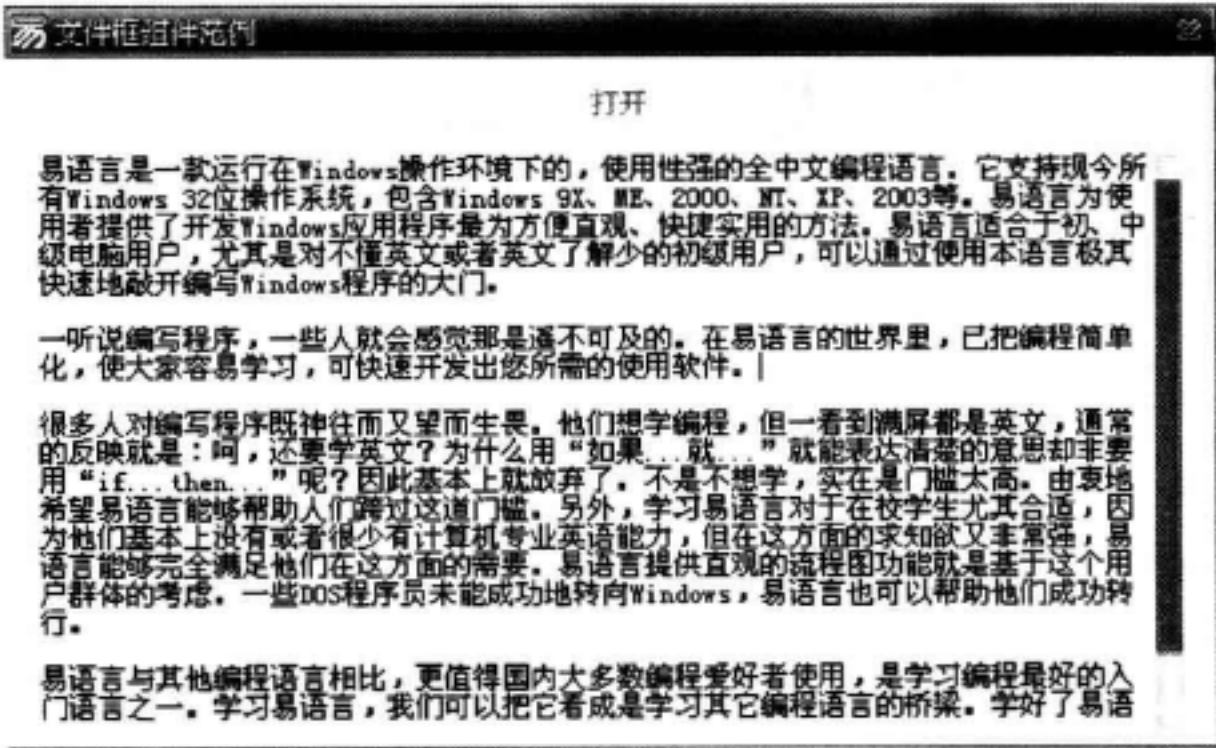



图 10-24 文件框组件范例运行结果

【范例解析】在此范例中, 定义了“_启动窗口. 标记”作为一个全局变量使用, 在启动窗口和打开窗口之间传递变量。在打开窗口中选定一个 txt 文件, 将其完整路径传递到启动窗口, 并将文件内容显示在启动窗口的编辑框中。

其中打开窗口程序集的最后一个双击选择事件子程序可以在双击文件框组件中的文件时直接打开所选择的文件。

10.6 端口组件

端口组件的主要功能是可以通过端口与其他机器设备进行通信,或控制其他机器设备的运行。

10.6.1 端口组件的属性

端口组件的属性很多都是其独有的属性,具体见图 10-25 所示。

1. “端口号”属性

端口号从 1 到 16,代表 COM1 到 COM16 端口。COM 表示串口,也就是平常所说的鼠标口,共有 9 根针。

2. “波特率”属性

该属性指定通信端口所使用的波特率,比较常用的波特率有: 110, 300, 600, 1200, 2400, 4800, 9600, 14400, 19200, 38400, 56000, 57600, 115200, 128000, 256000。

一般情况下,两个用于通信的机器设备要求波特率要一致。

打开系统的控制面板,双击“系统”图标,在弹出的系统属性对话框中选择“硬件”选项夹,点击“设备管理器”按钮,就进入设备管理器窗口,可以看到所有的硬件设备。选择其中的“端口”→“通讯端口(COM1)”,就可以对这个端口进行管理了,如图 10-26 所示。

端口1 (端口)	
名称	端口1
备注	
左边	256
顶边	96
宽度	24
高度	32
标记	
端口号	1
波特率	19200
数据位数	8
停止位数	1
奇偶校验	假
事件字符	
等待时间	1000
自动启动	假
奇偶校验方案	无
流控制方案	无
Rts流控制	默认
Dtr流控制	默认

图 10-25 端口组件的属性

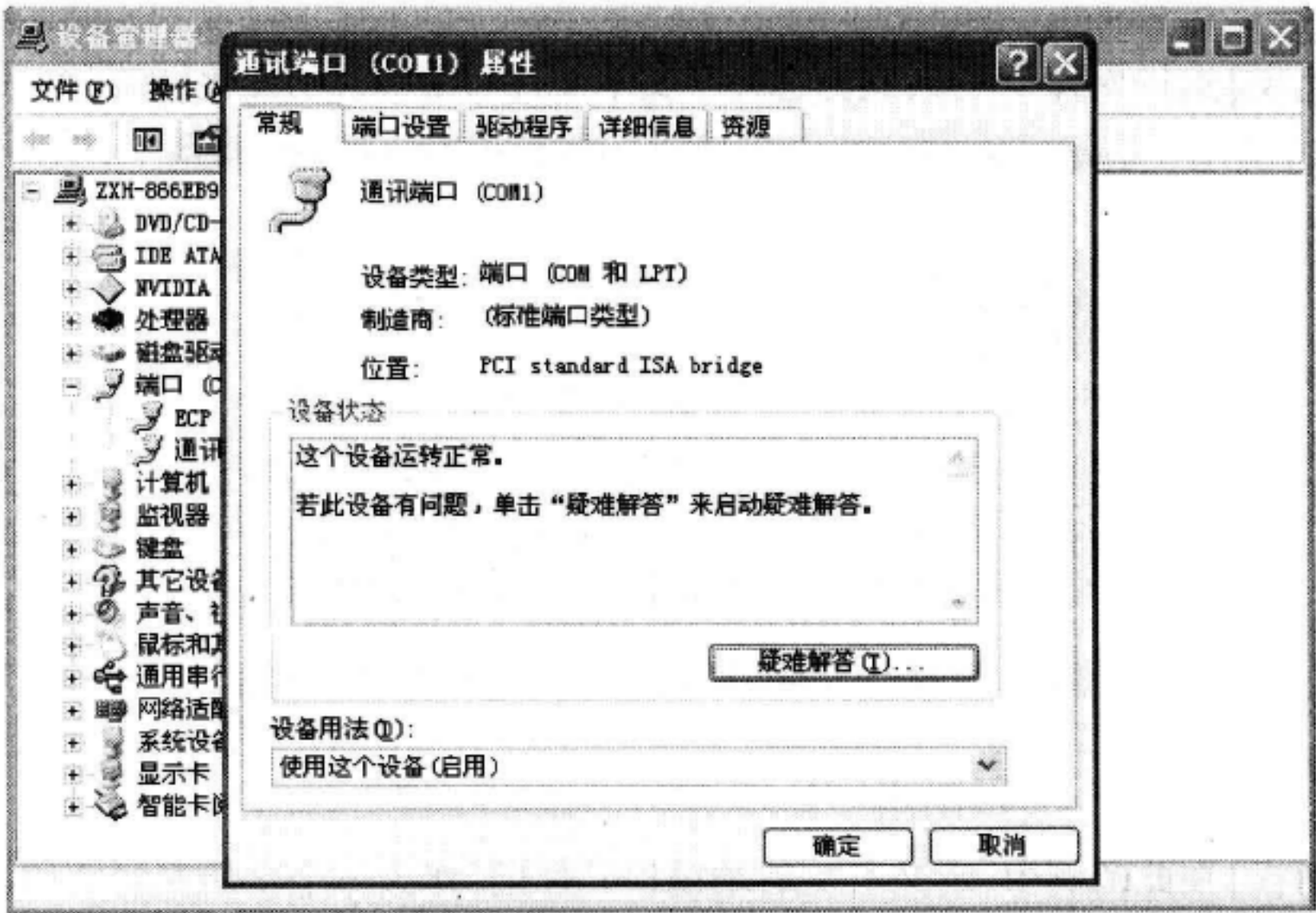


图 10-26 通信端口属性窗口

单击“端口设置”选项夹,可以对波特率等内容进行设置,如图 10-27 所示。



图 10-27 设置端口属性

3. “数据位数”属性

该属性的数据类型为整数型,可以为以下值之一:4、5、6、7、8。数据位数一般不用改。

4. “停止位数”属性

该属性的值可以为以下值之一,1:1 位、2:1.5 位、3:2 位。

5. “奇偶校验”属性

此属性指定通信时对数据是否进行奇偶校验。

6. “事件字符”属性

此属性指定用作触发事件的字符。这里若输入多个字符则只使用文本的首字符,输入为空则表示不支持事件字符。

7. “等待时间”属性

此属性指定在进行端口读写操作时的最长等待时间,单位为毫秒。如果超过等待时间,将返回错误信息。

8. “自动启动”属性

在对端口进行操作之前必须先启动,本属性指定当组件被载入时是否自动启动。

9. “奇偶校验方案”属性

此属性指定通信时所使用的具体奇偶校验位方案,其值可以为以下 5 种之一:(0)无、(1)奇校验、(2)偶校验、(3)标志校验、(4)空白校验。

10. “流控制方案”属性

此属性用于指定通信时所使用的流控制方案。

11. “Rts 流控制”属性

此属性指定通信时是否启用 Rts 流控制,其值可以为以下值之一:(0)默认、(1)禁止、(2)启用。

12. “Dtr 流控制”属性

此属性指定通信时是否启用 Dtr 流控制,其值可以为以下值之一:(0)默认、(1)禁止、(2)启用。

10.6.2 端口组件的事件

端口组件的事件有两个:收到信号、数据到达。

1. “收到信号”事件

当通信端口的信号状态发生改变时,会产生本事件。

2. “数据到达”事件

当通信端口接收到一个数据字节时,就会产生本事件。

10.6.3 端口组件的方法

端口组件的方法有:启动()、停止()、发送数据()和信号操作()。

1. “启动()”方法

该方法的功能是启动或重新启动对指定端口的操作。在对端口进行操作之前必须首先启动,如果在端口启动后又更改了端口属性必须重新启动。该方法成功返回“真”,失败返回“假”。其代码为:

端口 1. 启动()

2. “停止()”方法

该方法的功能是关闭已经启动的指定端口。其代码为:

端口 1. 停止()

3. “发送数据()”方法

该方法的功能是从端口发送指定的数据,成功返回“真”,失败则返回“假”。本命令为初级对象成员命令。

该方法的参数名称为“欲发送数据”,类型为“通用型(all)”,欲发送数据必须是系统基本数据类型。程序代码如下:

端口 1. 发送数据(发送编辑框. 内容)

4. “信号操作()”方法

本命令可以设置或清除通讯端口上指定信号的状态,成功返回“真”,失败返回“假”。本命令为初级对象成员命令。其调用格式为:

〈逻辑型〉对象. 信号操作(操作类型,欲操作信号类型)

(1) 参数“操作类型”的类型为“整数型(int)”。本参数可以为以下常量值之一:1: #清除信号;2: #发送或置位。

(2) 参数“欲操作信号类型”的类型为“整数型(int)”。本参数可以为以下常量值之一:1: #DTR 信号;2: #RTS 信号;3: #Break 信号。

【范例 10-6】在两个端口间进行端口通信测试。启动窗口中添加端口组件,添加两个按钮控制本端口的启动和停止,添加一个“发送”按钮,用来测试端口能否成功发送数据。设计窗口如图 10-28 所示。具体参考例程 10-6。

启动窗口的程序代码如图 10-29 所示。

【运行结果】运行例程 10-6,在对端口操作前必须先启动端口,然后在编辑框中输入数据,点击发送,若数据发送成功,运行结果如图 10-30 所示。

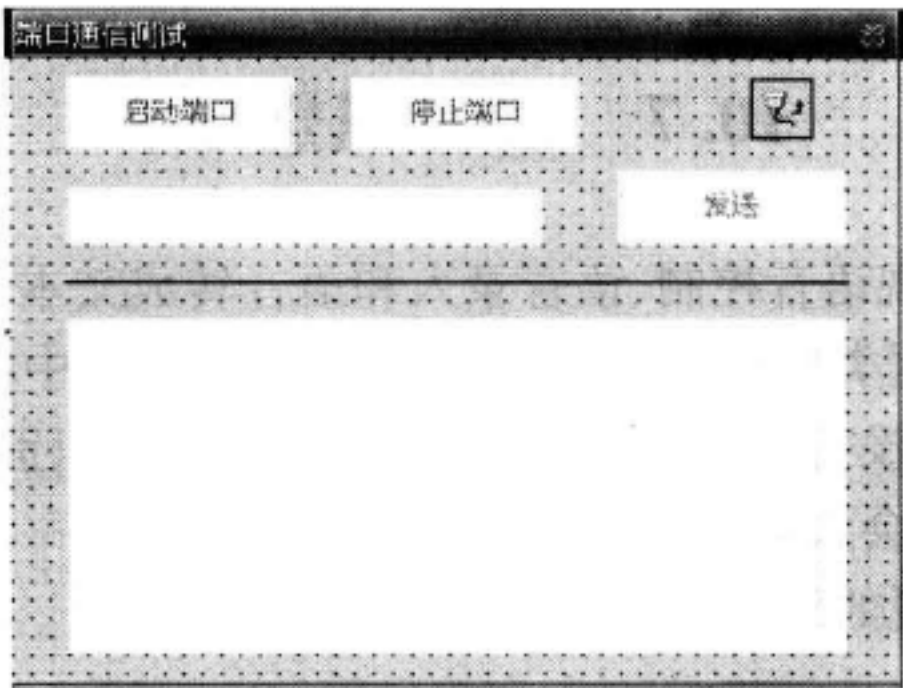


图 10-28 端口通信测试窗口

窗口程序集名	保留	保留	备注
窗口程序集1			

子程序名	返回值类型	公开	备注
_端口1_收到信号			

参数名	类型	参考	可空	数组	备注
信号类型	整数型				

接收编辑框, 加入文本 (“收到信号.” + 到文本 (信号类型) + #换行符)

子程序名	返回值类型	公开	备注
_端口1_数据到达			

参数名	类型	参考	可空	数组	备注
数据字节值	整数型				

接收编辑框, 加入文本 (“收到字节.” + 到文本 (数据字节值) + #换行符)

子程序名	返回值类型	公开	备注
_启动按钮_被单击			

如果真 (端口1.启动 () = 假)
信息框 (“启动失败”, 0,)

子程序名	返回值类型	公开	备注
_停止按钮_被单击			

端口1.停止 ()

子程序名	返回值类型	公开	备注
_发送按钮_被单击			

如果 (端口1.发送数据 (发送编辑框.内容) = 假)
信息框 (“发送失败”, 0,)
发送编辑框.内容 = “”
信息框 (“发送成功”, 0,)

图 10-29 启动窗口程序集

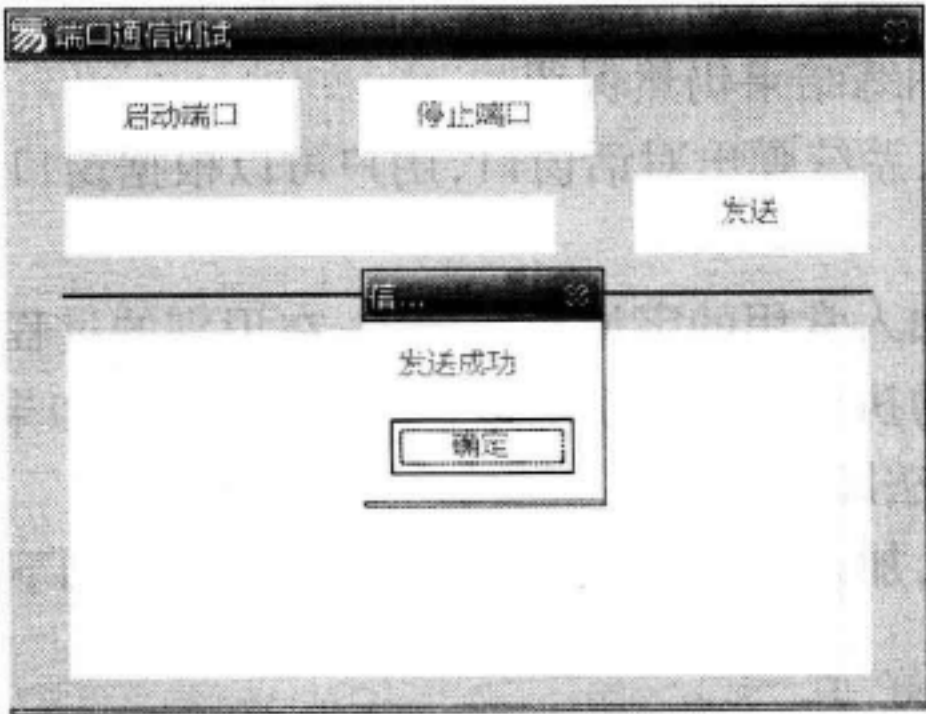



图 10-30 端口通信测试运行结果

10.7 语音识别组件

语音识别组件可实现语音控制、语音录入和语音转成文本等功能。可以使用语音识别组件制作类似语音输入数据类软件,即使用麦克便可以向电脑中输入内容了。当然因为每个人的发音状况不同,所以在没有进行任何处理的情况下,语音识别不够准确,但使用命令进行处理后,语音识别的准确度会大幅提高。

语音识别组件是不可视的组件,没有重要的属性。

10.7.1 语音识别组件的方法

语音识别组件的专有方法有4个:创建()、释放()、训练()、加入常用()方法。

1. “创建()”方法

该方法用于创建语音识别引擎,该引擎提供语音识别的功能,成功返回“真”,失败返回“假”。本命令为初级对象成员命令。该方法的调用格式为:

〈逻辑型〉对象.创建(整数型 识别引擎,整数型 作用域)

(1) 参数“识别引擎”的类型为“整数型(int)”。本参数应提供由“创建”命令所要求的识别引擎常量。该参数为0代表创建中文识别引擎,为1代表创建英文识别引擎。

(2) 参数“作用域”的类型为“整数型(int)”。本参数提供由“创建”命令所要创建的识别系统的作用范围常量。该参数为0代表系统作用域,指的是无论识别程序是否拥有系统焦点,该识别程序都能进行语音识别。该参数为1代表程序作用域,指的是只有在识别程序拥有系统焦点的时候,该识别程序才能进行语音识别。

【注意】创建一个与系统中安装的语音引擎的连接,如果系统中没有语音引擎,创建将会失败。如果系统中没有安装可以去微软的官方网站上获得。如果系统中已经安装 Microsoft Windows XP 或以上的版本,就不用再安装语音引擎组件包。

2. “释放()”方法

此方法用来释放已经创建的语音识别引擎。当不需要使用语音识别功能时,可以使用该方法,将创建的语音识别引擎释放,以释放内存空间。如果想继续使用语音识别引擎,则需要重新创建。

3. “训练()”方法

通过训练可以提高系统语音识别引擎的识别精确度,训练越多,识别系统对训练人的语音辨认越好,退出程序后,该训练结果仍然保留。

“训练()”方法运行后,系统弹出对话框,用户可以根据窗口提示,一步步地进行训练。

4. “加入常用()”方法

该方法在识别系统中加入常用的字词或者句子,在识别的过程中,这些字词或者句子更加容易被系统识别。该方法的执行结果仅仅对该识别程序有效,如果要重新设置常用的字词或者句子,直接再次调用该方法即可。

例如,将几个常用的词,加入到语音识别常用语句中,代码如下:

加入成员(常用语句,“你好”)

加入成员(常用语句,“对不起”)

加入成员(常用语句,“再见”)
语音识别 1. 加入常用(常用语句)

10.7.2 语音识别组件的事件

“语音识别”组件只有一个事件:识别到语音。

当麦克接收到声音时,触发“识别到语音”事件。该事件有一个参数为“识别文本”,该文本是语音识别引擎对麦克接收到的声音进行识别后得到的文本。

【范例 10-7】语音控制运行指定程序,在启动窗口添加语音识别组件和其他相关组件,如图 10-31 所示。具体参考例程 10-7。



图 10-31 语音控制设计窗口

启动窗口的程序代码如图 10-32 所示。

子程序名	返回值类型	公开	备注
__启动窗口_创建完毕			

变量名	类型	静态	数组	备注
命令集	文本型			
命令数组	文本型		0	
临时数组	文本型		0	
临时命令	语音命令			
变量	整数型			

```
命令集 = "画板"&"_mspaint.exe@_计算器"&"_calc.exe@_记事本"&"_notepad.exe@_扫雷"&"_winmine.exe"
' 定义命令集,用@_分隔每条命令,用_分隔命令文本和命令代码
命令数组 = 分割文本(命令集, "@_", )
' 计次循环首 (取数组成员数 (命令数组), 变量)
列表框1.加入项目 (命令数组 [变量], 变量)
临时数组 = 分割文本 (命令数组 [变量], "_", )
临时命令.命令文本 = 临时数组 [1]
临时命令.命令代码 = 临时数组 [2]
加入成员 (命令, 临时命令)
加入成员 (常用语句, 临时数组 [1])
' 计次循环尾 0
```

子程序名	返回值类型	公开	备注
_语音识别1_识别到语音			

参数名	类型	参考	可空	数组	备注
识别文本	文本型				

变量名	类型	静态	数组	备注
变量	整数型			

```
' 计次循环首 (取数组成员数 (命令), 变量)
' 如果真 (命令 [变量].命令文本 = 识别文本)
运行 (命令 [变量].命令代码, 假, )
编辑框1.加入文本 (识别文本 + #换行符)
' 跳出循环 0
' 计次循环尾 0
```

图 10-32 启动窗口程序集

【运行结果】运行例程 10-7,观测当麦克风接收到不同语音信号时的变化,如图 10-33 所示。

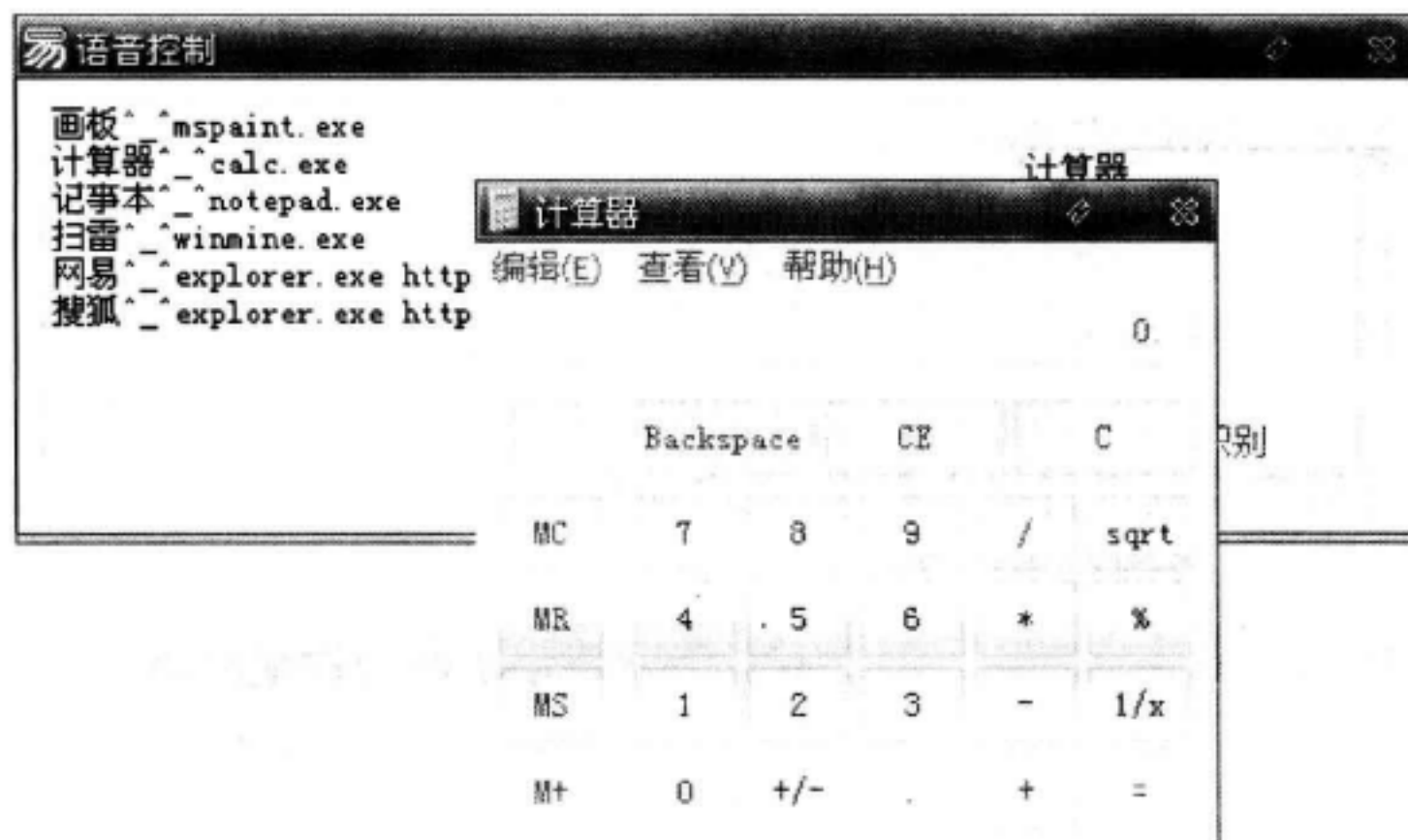


图 10-33 语音控制例程运行结果

【范例解析】首先将需要运行的程序名和命令名分别加入到 2 个数组变量中,其中将程序名数组加入到常用语句中,提高识别这些语句的准确性。然后在“识别到语音”事件子程序中编写代码,当接收到指定命令,则运行相应的程序。事件触发后则用循环判断当前的识别文本和命令变量中的哪个命令相符,如果识别文本为命令变量中的命令,则使用“运行()”命令,运行指定的程序。

10.8 电话控制组件

电话控制组件^①可以对电话设备进行控制,可以实现自动拨号、自动回复、动态语音提示、发送和接收传真等功能,可以很方便地实现电话自动语音提示查询系统等功能。

10.8.1 电话控制组件的属性

电话控制组件的重要属性有“线路总数”、“线路名称”、“线路号”和“线路类型”等属性。

1. “线路总数”属性

该属性为系统中可用的线路的总数。电话与计算机都是使用调制解调器连接的,该属性返回的是当前与计算机相连的可用的电话线路总数。

2. “线路名称”属性

该属性为当前的线路名称。计算机可以连接多部电话,即多个线路,每个线路都有默认的名称,该属性为当前线路的名称。

3. “线路号”和“线路类型”属性

“线路号”属性可以设置和返回当前的线路号,使用该属性,就可以在多个线路之间进行切换,线路号从 1 开始,最后一个线路号即为“线路总数”属性的值。

“线路类型”属性为设置和返回当前线路的类型,有 3 种取值情况:1. 电话、2. 主机、3. IP。

10.8.2 电话控制组件的方法

电话控制组件的方法比较多,具体如下:

1. “初始化()”和“清除()”方法

“初始化()”方法用来初始化电话语音支持库。使用“电话控制”组件的方法前必须调用此方法。所以,此方法适合在“_启动窗口创建完毕”事件子程序中运行。

“清除()”方法与“初始化()”方法对应,清除电话语音支持库所有资源。清除后,要想继续使用“电话控制”组件,必须重新初始化。

2. “呼叫()”方法

“呼叫()”方法的功能是用当前线路呼叫指定终端。终端类型包括电话号码、IP 地址、主机名。其调用格式为:

〈逻辑型〉电话控制. 呼叫(终端号码,等待时间)

- 参数“终端号码”为文本型,指的是对方的电话号码或 IP 地址或主机名。
- 参数“等待时间”为整数型,是指要等待的时间(秒),如果超出时间对方无应答则自动挂断。默认为 60 秒,最长等待时间为 120 秒。

【注意】第一个参数如果选择填写 IP 地址或主机名,需要保证对方机器上也连有电话机。

“呼叫()”方法相当于打电话时的拨号过程,成功呼叫后,终端的电话会响铃。如果终端也运行了有“电话控制”组件的程序,则终端会触发“振铃”事件,在“振铃”事件子程序中,可以使用“应答()”方法来应答呼叫。

3. “断开()”、“应答()”方法

“断开()”方法用来断开当前的呼叫,即相当于打电话时的挂断电话。

“应答()”方法用来应答呼叫,即接听来电。本方法必须在“振铃”事件后调用。

4. “播放()”方法

该方法用来播放一个语音文件,放音完毕后会产生产生“放音完毕”事件。其调用格式为:

〈逻辑型〉电话控制. 播放(文件名,中断)

- 参数“文件名”的类型为文本型,为要播放的文件名称,如果不指定路径则为当前目录。
- 参数“中断”的类型为整数型,表示是否可以按键中断播放,可以是“按键中断”数据类型中的常量。

当程序播放语音文件时,来电方就会在电话中听到语音,可以使用该方法来播放自动语音提示,来提示来电用户如何进行操作,可以在“放音完毕”事件子程序中来判断用户选择了哪种操作。

【注意】播放的文件必须是 *.WAV 文件。

5. “录音()”方法

该方法可以记录来电方的声音,录音完毕后会产生产生“录音完毕”事件。其调用格式为:

〈逻辑型〉电话控制. 录音(文件名,中断,时间)

- 参数“文件名”的类型为文本型,为存放录音文件的名称,如果不指定路径则为当前目录。
- 参数“中断”的类型是整数型,指是否可以按键中断播放,可以是“按键中断”数据类型中的常量。
- 参数“时间”的类型是整数型,为录音的时间(秒)。可以使用该方法实现语音留言等功能。

6. “停止()”方法

该方法可以停止当前的播放或录音。

7. “产生按键()”方法

该方法在当前线路产生一串按键,等同于在话机上按键。有一个参数,包括一系列按键的字符串。可用的字符包括 1、2、3、4、5、6、7、8、9、0、*、# 以及英文的逗号“,”,逗号的作用是延迟两个相邻按键之间的间隔。

8. “保持()”方法

该方法用来保持当前的呼叫状态。有一个参数,为“真”则保持呼叫状态,为“假”则解除保持。“呼叫()”方法当超过一定时间后会自动挂断,用“保持()”方法可以保持呼叫的状态。

9. “收集按键()”方法

该方法的功能是收集一串按键,收集按键完毕后会产生产生“收集按键完毕”事件。其调用格式为:

〈逻辑型〉电话控制. 收集按键(按键个数,终止按键,按键时间 1,按键时间 2)

- 参数“按键个数”的类型为整数型,表示要收集的按键的个数。
- 参数“终止按键”的类型为字节型,表示终止收集的按键。
- 参数“按键时间 1”的类型为整数型,等待第一个按键所用的最长时间(秒)。
- 参数“按键时间 2”的类型为整数型,代表各按键之间的最大时间间隔(秒)。

当前有很多种类的电话自动语音提示系统,如电话查询考试成绩系统、电话查询手机话费系统等,这些系统中都要求输入考号、账号、手机号等信息,并按“#”号表示结束,这个功能就可以使用“收集按键()”方法来实现。例如,输入一个 12 位的账号后按“#”号结束,可以使用代码:

电话控制. 收集按键(12,35,20,20)

当收集按键完毕后会产生产生“收集按键完毕”事件,该事件有 2 个参数,其中“按键串”参数记录了收集到的按键文本,可以判断该文本是否符合现有账号。

10. “设置声音文件()”、“播放数字()”方法

“设置声音文件()”方法用来设置声音文件,为“播放数字()”方法做准备。

“播放数字()”方法,根据“设置声音文件()”方法中设置的文件或以支持库默认的声音播放指定的数字。

“播放数字()”方法的调用格式为:

〈逻辑型〉电话控制. 播放数字(数字,播放类型,声音类型,间隔时间)

- 参数“数字”的类型为双精度小数型,本参数用来表示要播放的数字。
- 参数“播放类型”的类型为整数型,可为“播放数字类型”数据类型中的常量值。
- 参数“声音类型”的类型为整数型,1 为自定义,2 为默认女声,3 为默认男声。如为 1 (自定义),要在调用此方法之前调用“设置声音文件”来设置自定义声音。

- 参数“间隔时间”的类型为整数型,为每播放一个数字中间间隔的时间(秒)。

在没有特殊需要时,完全可以使用默认的声音,例如,朗读数字 2 的代码如下所示:

电话控制. 播放数字(2,#播放数字类型. 数字,2,1)

11. “初始化传真()”、“清除传真()”、“发送传真()”、“开启接收传真()”方法

这四个方法是用来控制传真功能的,需安装 Windows 的“传真服务”附件。

“初始化传真()”方法的功能是初始化传真设备。有一个参数为系统传真的设备号,默认

为1,即是系统的第一个传真设备。在使用传真功能前,一定要使用本方法对传真设备进行初始化。

“清除传真()”方法与“初始化传真()”方法对应,用于清除所有传真资源。

“发送传真()”方法用来发送传真。其调用格式为:

〈逻辑型〉电话控制.发送传真(电话号码,文件名)

- 参数“电话号码”的类型为文本型,本参数为对方的电话号码。
- 参数“文件名”的类型为文本型,本参数为要发送的文件。

“开启接收传真()”方法可以设置当接收传真开启后有电话打过来时自动接收传真。参数为逻辑型,“真”为开启,“假”为关闭。

例如:开启本程序传真接收功能并发送一个文本传真的代码如下:

电话控制.初始化传真(1)

电话控制.开启接收传真(真)

电话控制.发送传真(88995831,取运行目录()+“\下月计划.txt”)

电话控制.清除传真()

【注意】如果程序由始至终都要使用传真功能,可以在启动窗口创建完毕后即初始化传真和开启接收传真,当程序即将销毁时再清除传真。

10.8.3 电话控制组件的事件

电话控制组件使用时,一般都在其事件处理子程序中做相关处理。下面详细介绍电话控制组件的常用事件。

1. “接通”事件

当线路接通时触发此事件。可以在该时间中播放欢迎信息等。

2. “振铃”事件

当有电话呼入的时候每次振铃都将产生此事件,出现振铃事件后就可以调用“应答()”方法应答呼叫。

3. “来电显示”事件

当来电号码被捕获的时候,将产生此事件。触发该事件的前提是必须保证本机的电话具有来电显示功能。该事件有1个参数,即为捕获到的来电号码。

4. “断开”事件

当呼叫被断开时,将产生此事件。该事件中可以处理一些扫尾工作,例如将保存当前来电用户信息的变量清空等。

5. “检测按键”事件

当通话一方按下话机上的按键时产生此事件。可以用“检测按键”方法来启动或关闭检测按键事件。

6. “收集按键完毕”事件

使用“收集按键()”方法开始收集按键,当收集完毕则触发该事件,该事件有2个参数,第一个参数为收集到的按键文本;第二个参数为“事件触发原因”,该参数值的意义如下:1表示收集的按键已达到了指定的个数,2表示指定的用于结束收集的按键被检测到了,3表示在等待第一个按键时超时,4表示在等待第一个以后的按键时超时。

7. “播放完毕”事件

当播放语音完毕后产生此事件。该事件有 1 个参数为播放完毕的文件名,可以判断播放完毕文件来进行不同的操作。

8. “录音完毕”事件

当录音完毕后产生此事件。

9. “保持”事件

当前呼叫的状态变为保持时触发此事件。可以在该事件子程序中判断何时来断开正在保持的呼叫。

10. “振铃之前”事件

当振铃之前触发此事件,可以在此事件用“开启接收传真”方法接收传真。

【范例 10-7】设计一个电话自动答录系统。在启动窗口中添加一个电话控制组件,如图 10-34 所示,要实现的功能均由代码实现,下面会一一介绍。具体参考例程 10-8。

【范例解析】本系统中,当振铃次数超过 1 次,则使用“应答()”方法接听来电,如果应答失败则使用信息框查看失败原因并返回;如果应答成功,则播放欢迎语音,语音内容是:“按 1 键为查询费用,按 2 键为录音”,完毕后运行“检测按键()”方法。实现以上功能的程序代码如图 10-35 所示。

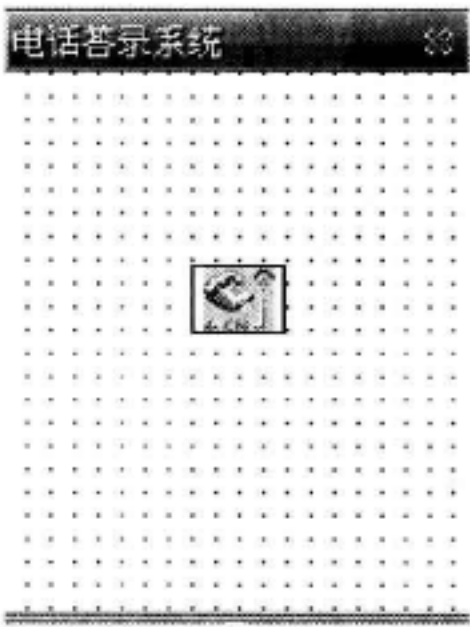


图 10-34 电话答录系统设计窗口

子程序名	返回值类型	公开	备注		
电话控制_振铃					
参数名	类型	参考	可空	数组	备注
次数	整数型				
如果真 (次数 < 1)					
返回 0					
* 如果真 (取反 (电话控制.应答 ()))					
信息框 (电话控制.错误信息, 0,)					
返回 0					
电话控制.播放 (#欢迎语音, #按键中断.任意键中断)					
电话控制.检测按键 (真)					

图 10-35 启动窗口程序集 1

“检测按键()”方法检测用户按下了电话中的哪个键,在“检测按键”事件子程序中会根据不同按键进行不同操作。当检测到对方按键后,则将“检测按键”事件关闭,然后判断对方按下的键并进行相关操作。程序中利用“检测按键层数”变量来记录程序是第几次检测按键,如果是第一次检测按键则表示是“欢迎语音”播放后进行的按键检测,键值 49 代表 1 键、50 代表 2 键。所以“检测按键层数”为 1 时,如果按下 1 键,则播放“输入电话号码语音”并收集对方输入的电话号码;如果按下 2 键,则播放“录音提示”。实现以上功能的程序代码如图 10-36 所示。

收集按键完毕事件中,判断“检测按键层数”变量为 1,则表示当前收集的按键为电话号码,首先在数据库中查找“电话号码”字段是否有当前输入的电话号码,如果找到则播放“输入密码语音”来提示用户开始输入密码;如果没找到当前输入的电话号码,表示用户输入错误,则再次播放输入“输入电话号码语音”提示用户重新输入电话号码。

判断“检测按键层数”变量为 2,则表示当前收集的按键是用户输入的密码,由于在判断用户输入电话号码是否正确时,已经将数据库的当前记录指针跳到用户输入的电话号码记录上了,所以这里只需读出当前记录的“密码”字段内容,然后跟用户输入的密码比较,如果输入的

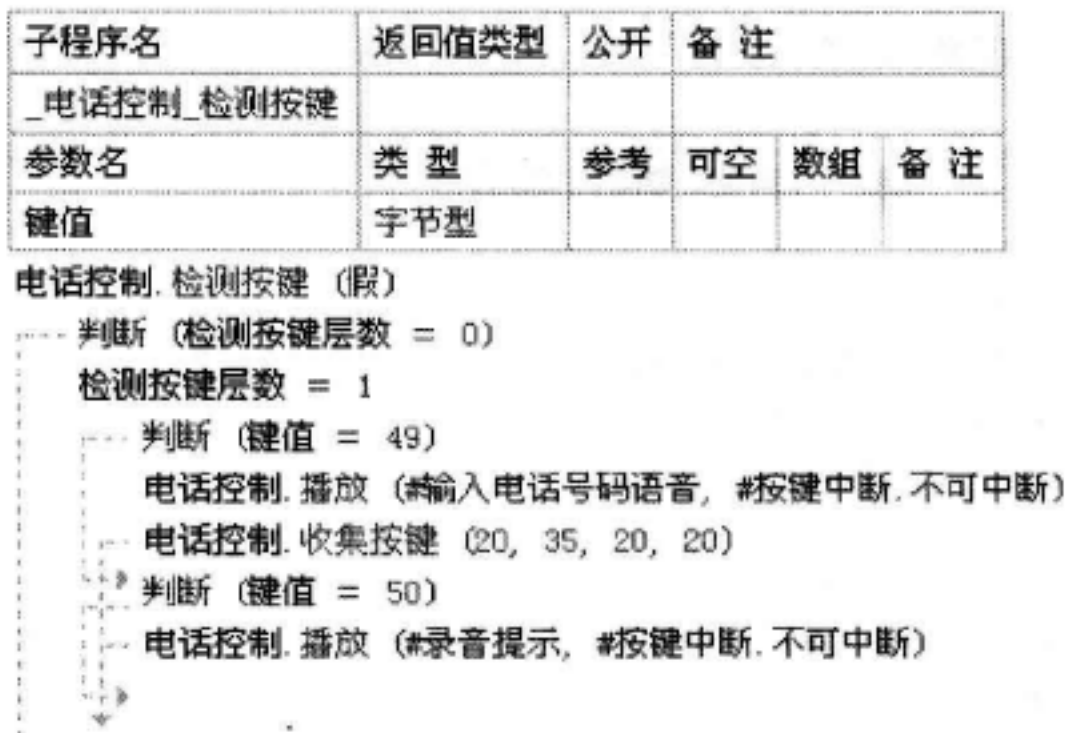


图 10-36 启动窗口程序集 2

密码错误,则播放“输入密码语音”来提示用户重新输入密码;如果输入的密码正确,则开始播放用户的账户余额。实现以上功能的程序代码如图 10-37 所示:

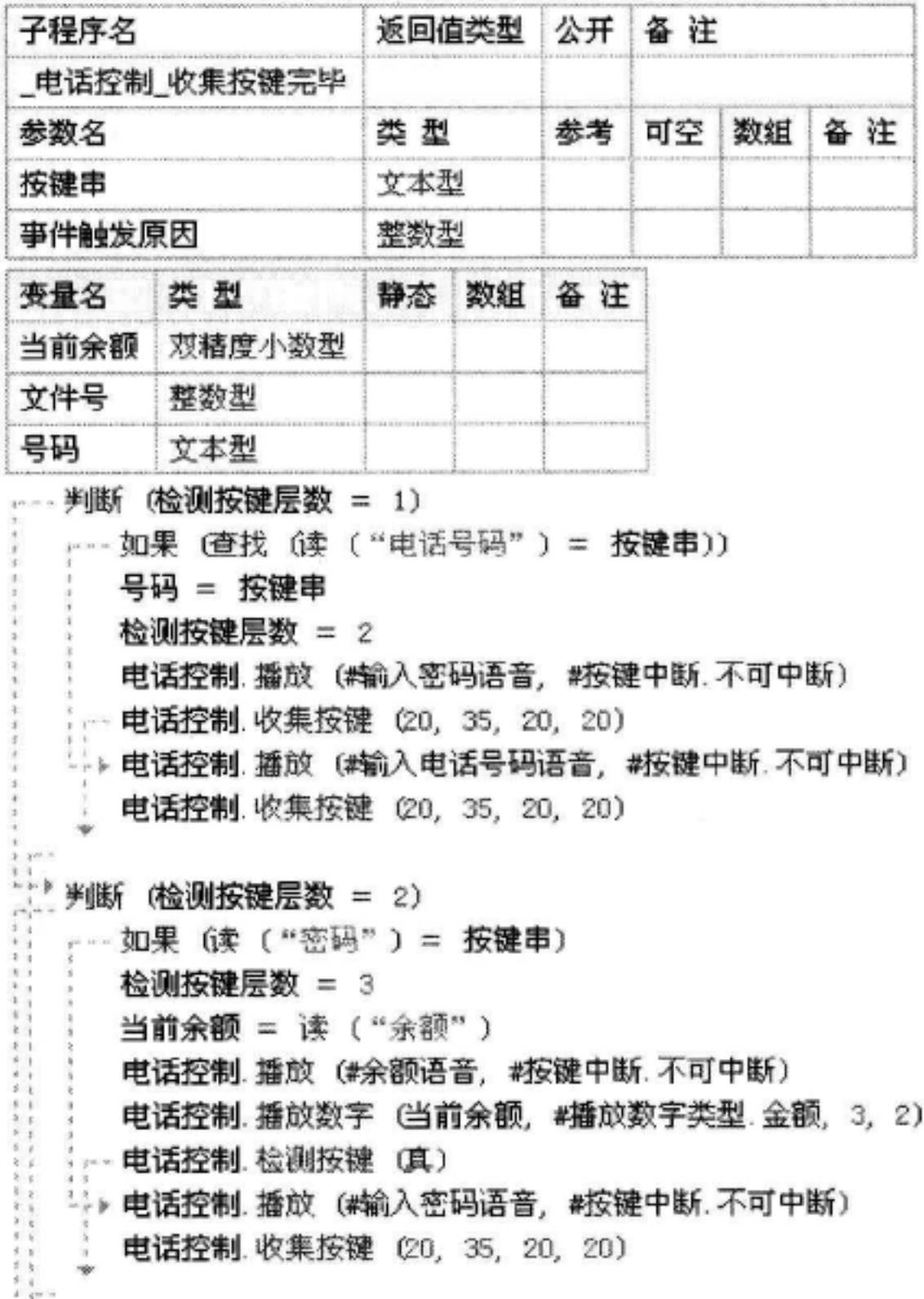


图 10-37 启动窗口程序集 3

系统中,根据播放完毕的不同文件进行不同的操作,如判断播放完毕的是“录音提示”则开始进行录音。实现本功能的程序代码如图 10-38 所示。

子程序名	返回值类型	公开	备 注		
电话控制_播放完毕					
参数名	类 型	参 考	可 空	数 组	备 注
文件名	文本型				


```
判断 (文件名 = #录音完毕)
电话控制.播放 (录音文件路径 + 录音文件名 + “.wav”, #按键中断.不可中断) ’ 播放录音文件
判断 (文件名 = 录音文件路径 + 录音文件名 + “.wav”)
电话控制.播放 (#查询结束, #按键中断.不可中断) ’ 查询结果
判断 (文件名 = #录音提示)
录音文件名 = 时间到文本 (取现行时间 (), )
如果真 (取反 (电话控制.录音 (录音文件路径 + 录音文件名 + “.wav”, #按键中断.不可中断, 10)))
信息框 (电话控制.错误信息, 0, )
```

图 10-38 启动窗口程序集 4

第 11 章 图片组组件

易语言中的图片组组件包括工具条、状态条、树形框、超级列表框,它们都有一个共同的特点,即有一个图片组属性,图片组组件与图片的展示有很大的关系,因此图片的制作技巧很重要。在很多情况下,一个程序界面漂亮与否也反映了交互性界面友好程度,如果界面很难看,即使功能再强大,也很难为用户所接受。

11.1 工具条组件

工具条组件通常置于窗口的上方,可以实现用户常用功能的最小化整合。我们经常使用的应用程序,如 IE 浏览器、office 办公软件等,它们的工具栏上都使用了工具条组件,如图 11-1 所示。

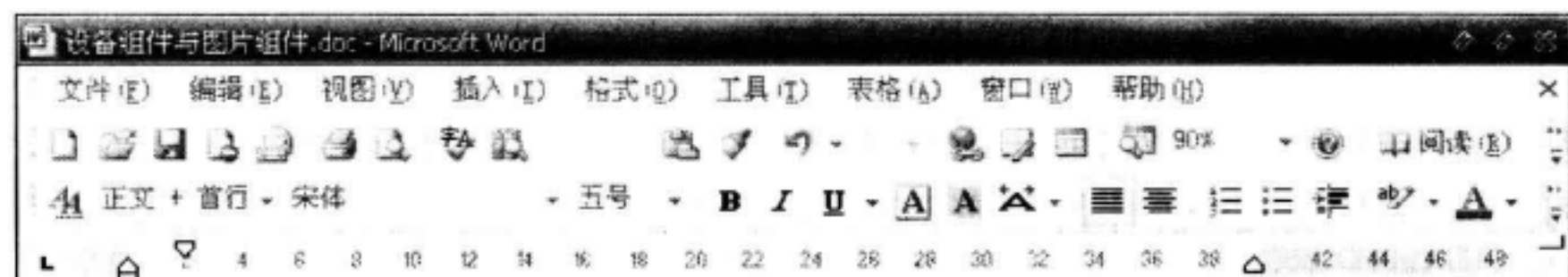


图 11-1 WORD 使用的工具栏

11.1.1 工具条组件的属性

工具条组件的重要属性有:位置、平面、上边线、列表方式、图片组、热点图片组、禁止图片组、工具条按钮等。具体如图 11-2 所示。

1. “位置”属性

本属性指定工具条的显示位置。它有 3 个可选项:“0. 顶边”、“1. 底边”、“2. 自由”。默认为“0. 顶边”。一般情况下,工具条都是显示在顶边的,只有极个别程序是将工具条显示在其他地方。

2. “平面”属性

本属性指定按钮是否为平面方式,此属性为逻辑型,默认为“真”。平面与立体只是按钮的外观发生了变化,对于其他功能操作方面没有影响。

【注意】分隔条只有在“平面”属性为“真”时才显示竖线出来,否则只是空一段。

3. “上边线”属性

本属性指定是否在工具条的顶部显示上边分隔线,此属性为逻辑型,默认为“真”。当属性为“真”时,会在菜单与工

工具条1 (工具条)

名称	工具条1
备注	
左边	0
顶边	0
宽度	374
高度	28
标记	
可视	真
禁止	假
鼠标指针	默认型
背景颜色	默认底色
位置	顶边
平面	真
上边线	真
列表方式	假
允许多行	真
起始空白	0
字体	
图片组	
热点图片组	
禁止图片组	
工具条按钮	

图 11-2 工具条组件的属性

具条之间加一横条线,使界面漂亮一些,对于其他功能操作方面没有影响。

4. “列表方式”属性

本属性指定在显示按钮时是否将标题文本显示在图片的右边,此属性为逻辑型,默认为“假”。

5. “图片组”属性

本属性为工具条各按钮提供标志图片。图片组是一排有统一宽度的 BMP 图片,当加入图片组后,即被自动按设定的像素点分割为带有序号的单一图片,以供工具条调用。

下面讲解制作此类图片的方法。

先打开一个有图标界面,使用抓图工具或使用键盘右上角的 PrScrn 键抓图,再运行 Photoshop,并在 Photoshop 中新建空白文件,然后将上述带有很多小图标的图片粘贴到这个空白文件中,如图 11-3 所示。



图 11-3 抓取用于参考的图片

放大其中的一个图标。使用矩形选择工具选择一个图标,接下来是查清它的大小,可以先复制,再使用 Ctrl + N 新建,这时新建空白图片的大小就是刚才复制的图标大小,如图 11-4 所示。本例中的图标宽度为 21 像素,高度为 20 像素。



图 11-4 新建空白文件

新建一个图标大小的文件后,用 Ctrl + V 键粘贴小图标到这个文件中,这样就形成有一个小图标的新图片文件。

只有一个图标不能满足程序的需要,若想连续放置多个图标就需要扩大图片的画布大小。以上图为例,在图片的标题栏上单击鼠标右键,在弹出的菜单中选“画布大小”,如图 11-5 所示。

例如,要制作 4 个图标的画布,就将宽度 21 个像素点乘以 4,得出的 84 个像素点即为新画

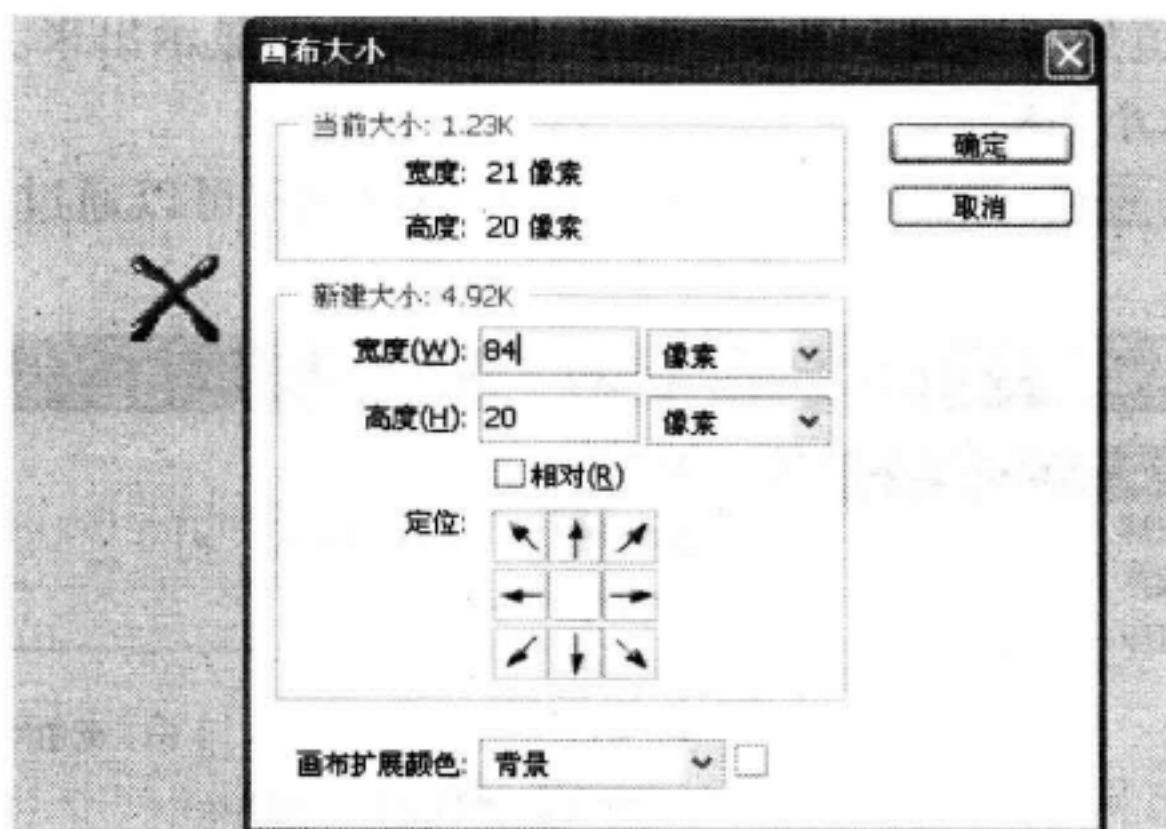


图 11-5 调整画布大小

布大小的宽度。然后,可以选择 4 个图标排列到新画布中,如图 11-6 所示。最后,将这张图片存盘为 BMP 文件格式即可由图片组载入。

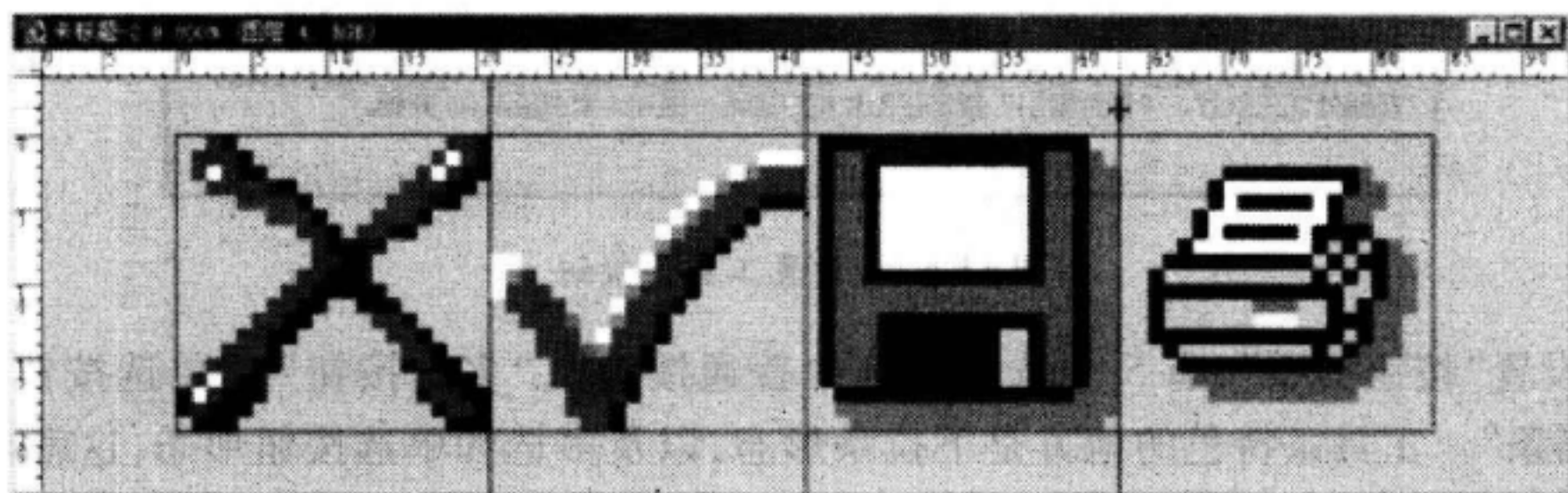




图 11-6 4 个图标

图片组加入内容可以参考以下步骤:

- (1) 点击图片组的浏览选择按钮,弹出“图片组设置”对话框,如图 10-35 所示。
- (2) 点击图片组设置对话框左下角的“加入”按钮,弹出“请指定欲加入图片并设置参数”对话框,如图 10-36 所示。
- (3) 在“单一图片宽度”后面输入像素数值,这是每个小图标的宽度,点击最上面的“图片文件名”后的按钮来浏览选择图片。

6. “热点图片组”属性

本属性为工具条各按钮提供当鼠标移动到此按钮上时所显示的标志图片,仅当“平面”属性为“真”时才有效。热点图片组的制作方法与图片组的制作方法一样。

7. “禁止图片组”属性

本属性为工具条各按钮提供当此按钮被禁止时所显示的标志图片。按下 Del 键即可删除现行内容。


【注意】工具条组件的“图片组”、“热点图片组”、“禁止图片组”3 种属性与图形按钮的“正常图片”、“点燃图片”、“禁止图片”属性类似。

8. “工具条按钮”属性

本属性设置工具条各按钮的信息。

在制作、载入图片组后,并没有在新的易程序中看到图标图片,这是因为虽然建立了图片

组,但由于系统还不知道如何排列小图标的顺序,因此就没有显示出来。通过设置本属性,将会按用户的要求来显示小图标。

单击本属性后面的按钮,弹出“工具条按钮设置”对话框,可以通过设置按钮属性来显示图标,如图 11-7 所示。

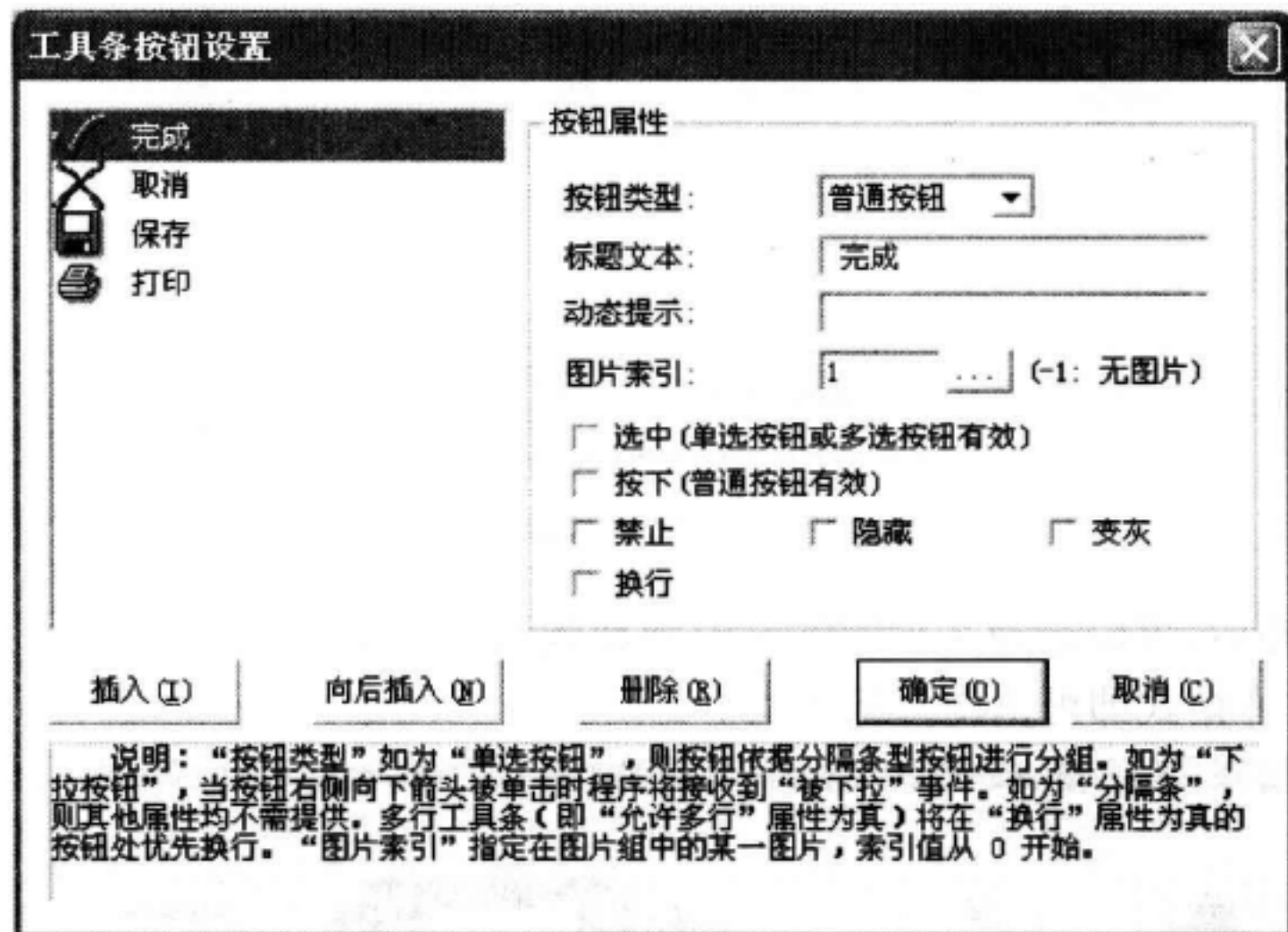


图 11-7 设置工具条按钮

通过设置“按钮类型”，有 5 个选项可选：“普通按钮”、“多选按钮”、“单选按钮”、“下拉条”、“分隔条”。工具条特色的地方是下拉条形态，以及多选和单选按钮形态，这就是比图形按钮组件优越的地方。

分隔条只有在“平面”属性为“真”时才显示竖线出来，否则只是空一段。

按钮属性中的其他设置，如“标题文本”是用于显示文字的；“动态提示”可以在鼠标移动到工具条时，显示一些说明文字；“图片索引”可以设置显示图片组中的哪一张图片。

【注意】为了使工具条按钮的文字显示宽松一些，可以在标题文本前后都加入空格。

11.1.2 工具条组件的事件

工具条组件的重要事件有：被单击事件和被下拉事件。

1. “被单击”事件

当工具条按钮被单击时即产生“被单击”事件。

2. “被下拉”事件

若工具条中包含下拉型按钮，当下拉型按钮右侧向下箭头被单击时即产生“被下拉”事件。

制作下拉按钮可以采用以下方式。

(1) 先要用菜单编辑器制作一个弹出菜单，根菜单是不可见的，如图 11-8 所示。

(2) 进入工具条被下拉的事件子程序，可以看到其中有 3 个参数，名为“按钮索引”的整数型参数，可以设置是哪一个按钮被击中，另两个参数可以知道被击中按钮的坐标。具体程序代码如图 11-9 所示。

运行后的效果如图 11-10 所示。

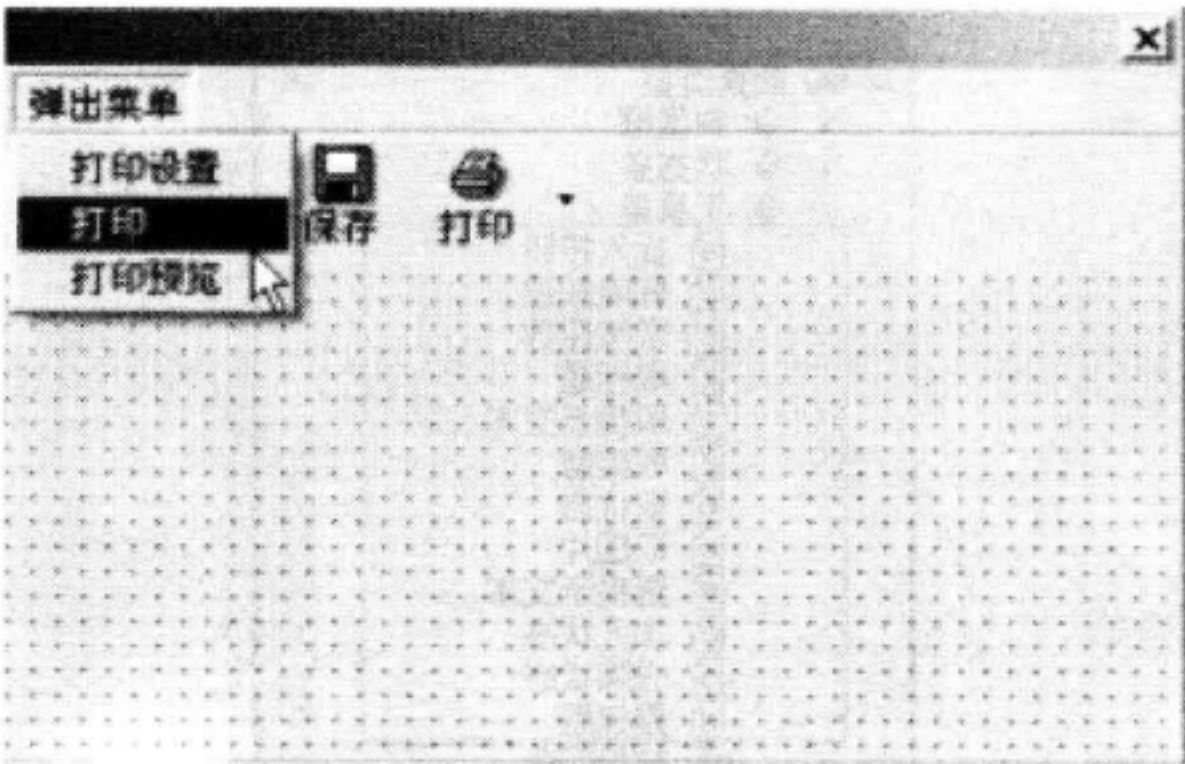


图 11-8 制作弹出菜单

子程序名	返回值类型	公开	备注		
工具条1_被下拉					
参数名	类型	参考	可空	数组	备注
按钮索引	整型				
下拉横坐标	整型				
下拉纵坐标	整型				

如果真 (删首尾空 (工具条1.取标题 (按钮索引)) = “打印”)
启动窗口.弹出菜单 (弹出菜单, 下拉横坐标 - 2, 下拉纵坐标 - 2)

图 11-9 工具条被下拉事件子程序

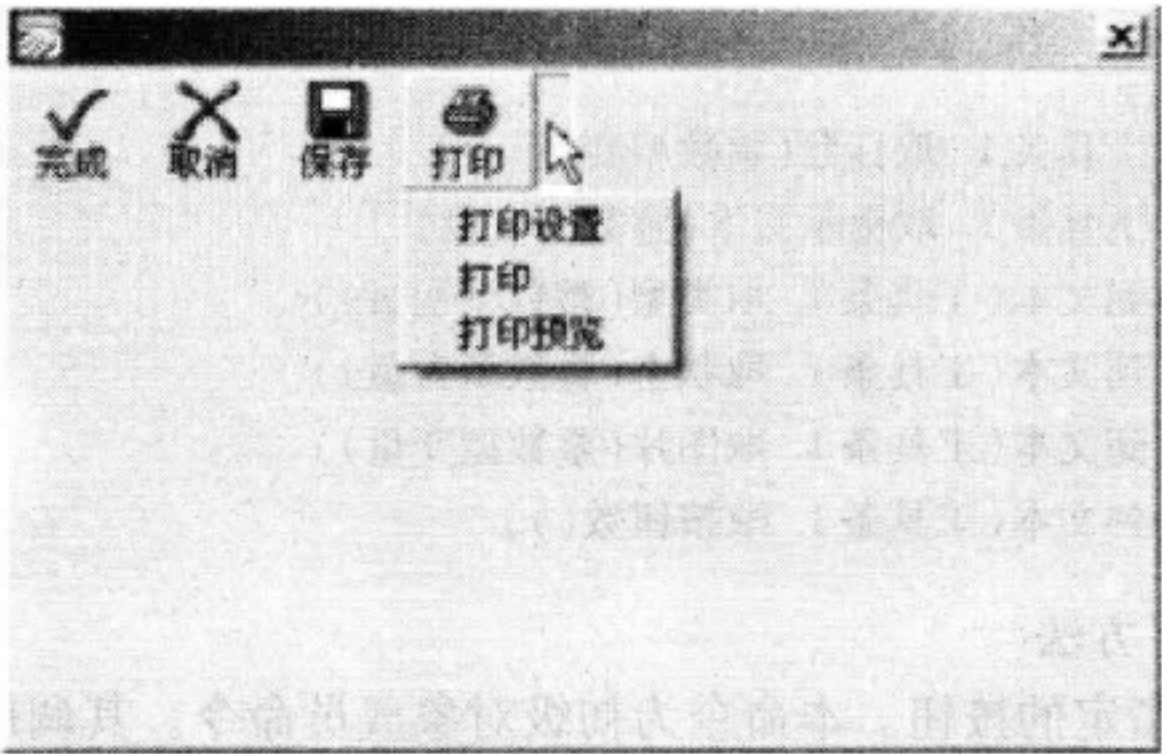


图 11-10 工具条下拉按钮运行效果

11.1.3 工具条组件的方法

工具条组件的专有方法可以在“扩展界面支持库一”中看到,如图 11-11 所示。

1. “取标题()”方法

此方法用于取回指定按钮的标题文本,本命令为初级对象成员命令。调用格式为:

〈文本型〉对象.取标题(按钮索引)

参数“按钮索引”的类型为“整型(int)”。0 为按钮一,1 为按钮二,以此类推。

其他的如“取提示文本”、“取类型”、“取状态”、“取图片”、“取按钮数”组件命令功能都类似。

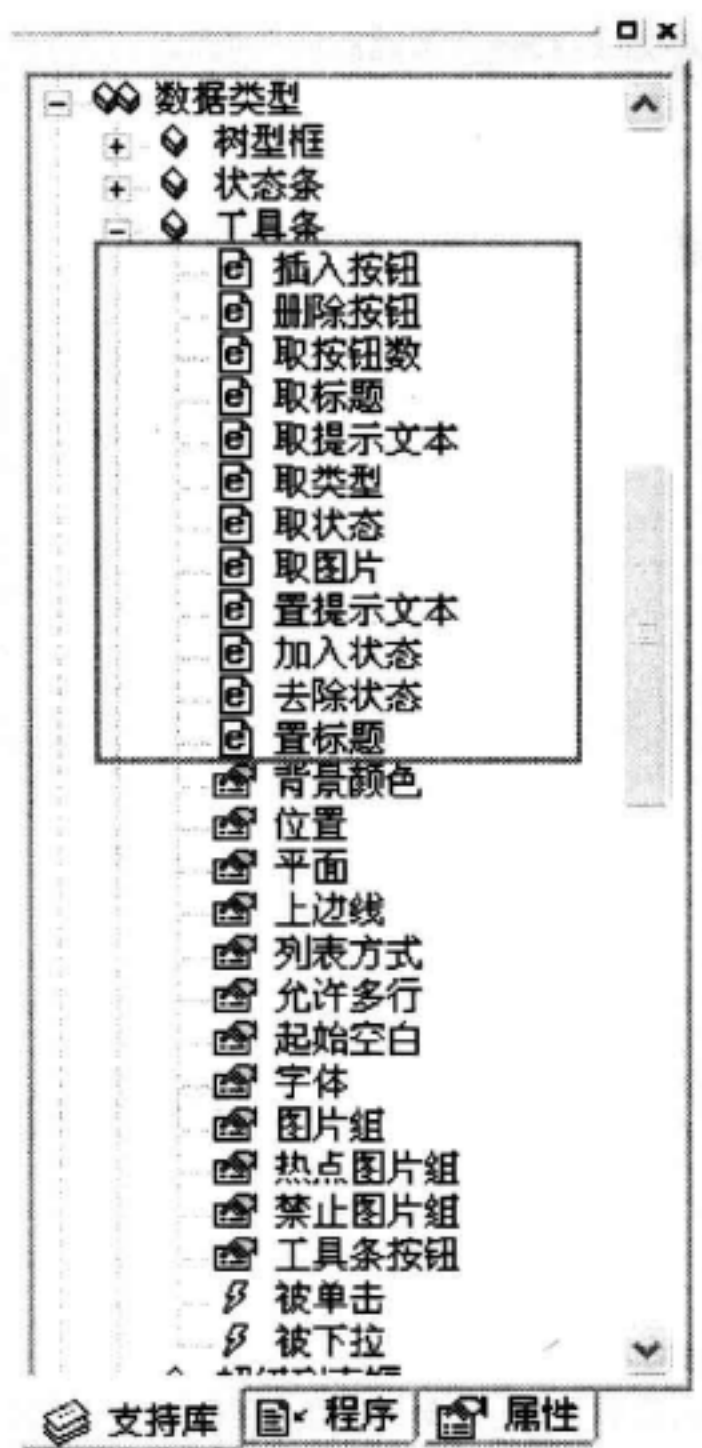


图 11-11 工具条的专有方法

程序代码如下所示：

```

编辑框 1. 内容 = 工具条 1. 取标题(整数型变量)
编辑框 2. 内容 = 工具条 1. 取提示文本(整数型变量)
编辑框 3. 内容 = 到文本(工具条 1. 取类型(整数型变量))
编辑框 4. 内容 = 到文本(工具条 1. 取状态(整数型变量))
编辑框 5. 内容 = 到文本(工具条 1. 取图片(整数型变量))
编辑框 6. 内容 = 到文本(工具条 1. 取按钮数())
  
```

2. “删除按钮()”方法

此方法用于删除指定的按钮。本命令为初级对象成员命令。其调用格式为：

〈无返回值〉对象. 删除按钮(整数型 按钮索引)

参数“按钮索引”的类型为“整数型(int)”。0 为按钮一, 1 为按钮二, 如此类推。

程序代码如下所示：

```
工具条 1. 删除按钮(0)
```

3. “加入状态()”、“去除状态()”方法

此方法用于将工具条变为禁止状态或非禁止状态, 程序代码如下：

```
子程序: 禁止或允许按钮_被单击
```

```
如果(位与(工具条 1. 取状态(0), #禁止) = #禁止)
```

```
    工具条 1. 去除状态(0, #禁止)
```

```
否则
```


工具条 1. 加入状态 (0, #禁止)

如果结束

4. “插入按钮()”方法

此方法可以在指定位置加上一个新按钮,调用格式为:

〈整数型〉对象.插入按钮([整数型 插入位置],[整数型 类型],[整数型 图片索引],[文本型 标题],[文本型 动态提示文本],[整数型 按钮状态])

参数“插入位置”的类型为“整数型(int)”,可以被省略。指定新按钮插入的位置索引,0 为栏目一,1 为栏目二,如此类推。如果提供 -1,则插入到工具条的尾部。如果本参数被省略,默认值为 -1。

参数“类型”的类型为“整数型(int)”,可以被省略。按钮的类型可以为以下常量值之一:0. #普通按钮、1. #多选按钮、2. #单选按钮、3. #下拉按钮、4. #分隔条。如果本参数被省略,默认值为“#普通按钮”。

参数“图片索引”的类型为“整数型(int)”,可以被省略。图片索引用于指定图片组属性中的某张图片,该图片将被显示在按钮表面。索引值从 0 开始,-1 表示无图片。如果本参数被省略,默认值为 -1。

参数“标题”的类型为“文本型(text)”,可以被省略。本参数指定显示在按钮下的标题文本。如果被省略,默认值为空文本。

参数“动态提示文本”的类型为“文本型(text)”,可以被省略。本参数指定当鼠标移动到按钮上并停留一段时间后所动态显示的提示文本。如果被省略,默认值为空文本。

参数“按钮状态”的类型为“整数型(int)”,可以被省略。本参数指定按钮的存在状态。状态值为 0 或者以下常量值任意组合(相加):1. #选中、2. #按下、4. #禁止、8. #隐藏、16. #变灰。其中“选中”状态在类型为多选按钮或单选按钮时有效,“按下”状态在类型为普通按钮时有效。如果本参数被省略,默认值为 0。

程序代码如下:

子程序: _禁止或允许按钮 1_被单击

工具条 1. 插入按钮(1, #普通按钮, 14, “停止”, “结束程序”, 0)

【范例 11-1】实现工具条的动态添加。首先在启动窗口通过工具条设置对话框添加四个工具按钮,然后再通过工具条的菜单编辑器对话框将“打印”按钮设置为下拉按钮,如图 11-10 所示。具体步骤前文中已经详细介绍过,此处不再赘述。最后添加一个按钮用于动态生成新工具条按钮,启动窗口如图 11-12 所示。具体参考例程 11-1。

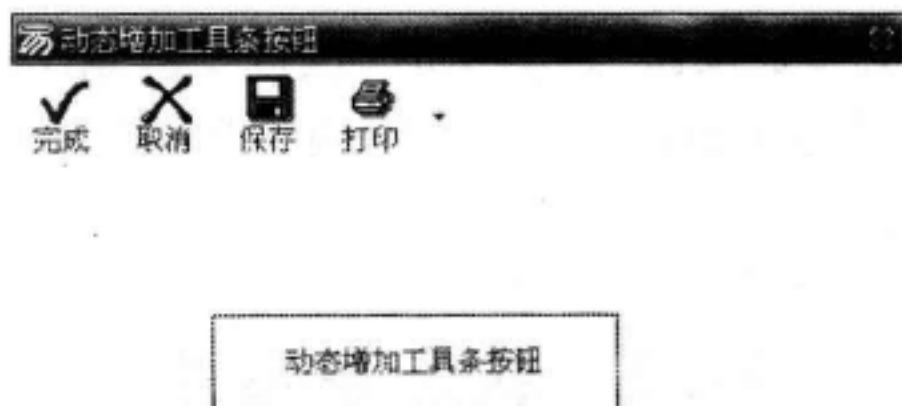


图 11-12 工具条例程设计窗口

启动窗口的程序代码如图 11-13 所示。

窗口程序集名	保留	保留	备注
窗口程序集1			

子程序名	返回值类型	公开	备注		
_工具条1_被单击					
参数名	类型	参考	可空	数组	备注
按钮索引	整数型				

信息框 (“按钮” + #左引号 + 工具条1.取标题 (按钮索引) + #右引号 + “被单击。”, 0,)

子程序名	返回值类型	公开	备注		
_工具条1_被下拉					
参数名	类型	参考	可空	数组	备注
按钮索引	整数型				
下拉横坐标	整数型				
下拉纵坐标	整数型				

如果真 (删首尾空 (工具条1.取标题 (按钮索引)) = “打印”)
_启动窗口.弹出菜单 (弹出菜单, 下拉横坐标 - 2, 下拉纵坐标 - 2)

子程序名	返回值类型	公开	备注
_按钮1_被单击			

工具条1.插入按钮 (3, 0, 1, “新增”, “新增加的图标!”,)

图 11-13 启动窗口程序集

【运行结果】运行例程 11-1,观测单击“动态增加工具条按钮”的变化,如图 11-14 所示。

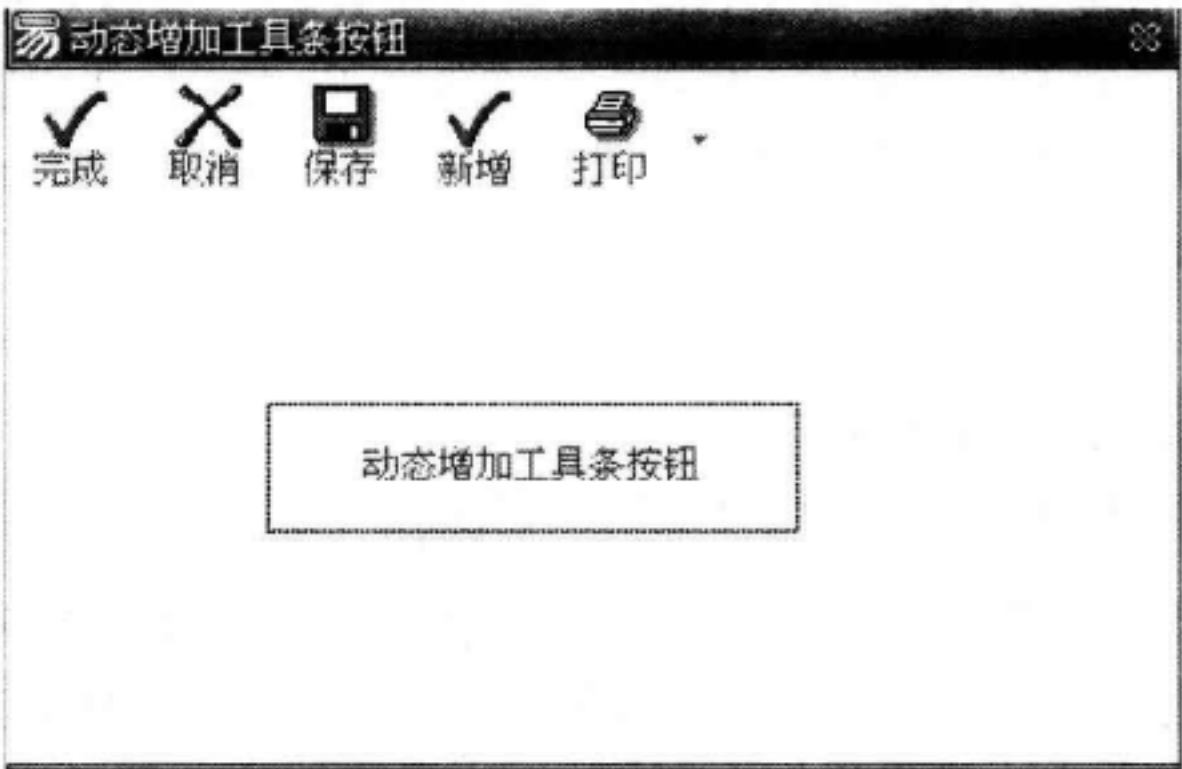


图 11-14 工具条例程运行结果

【范例解析】“动态增加工具条按钮”被单击时,执行工具条 1 的“插入按钮”方法,将一个普通按钮插入到工具条栏目号为 3 的位置,所使用的图片是在工具条按钮设置窗口中设置的图片索引为 1 的图片,标题为“新增”。

11.2 状态条组件

状态条组件一般处于程序的下方,起一个操作提示的功能。如 IE 浏览器的状态条可以显示当前操作的链接信息以及网页下载进度等,WORD 的状态条可以让用户知道当前的总

页数、当前页数、当前节数、当前行数、当前列数等信息,Photoshop 的状态条可以让用户知道缩放比例、运行文件大小、效率、工具使用提示等。

11.2.1 状态条组件的属性

1. “位置”属性

该属性指定工具条的显示位置。它有 3 个可选项:“0. 底边”、“1. 顶边”、“2. 自由”,默认为“0. 底边”。状态条一般都是在窗口的下方,因此选用“底边”较多。

2. “尺寸调节器”属性


该属性为“真”后,可在组件的右下方出现几条斜线,如图 11-15 所示。本属性如果在窗口的“边框”属性为非可调节选项时不起任何作用。



图 11-15 “尺寸调节器”属性为“真”与“尺寸调节器”属性为“假”

3. “项目”属性

该属性设置状态条各栏目的信息。

状态条组件的“项目”属性与工具条的“按钮”操作类似,只是栏目属性有一些改变。单击此属性后面的按钮后,弹出一个状态条栏目设置对话框,如图 11-16 所示。

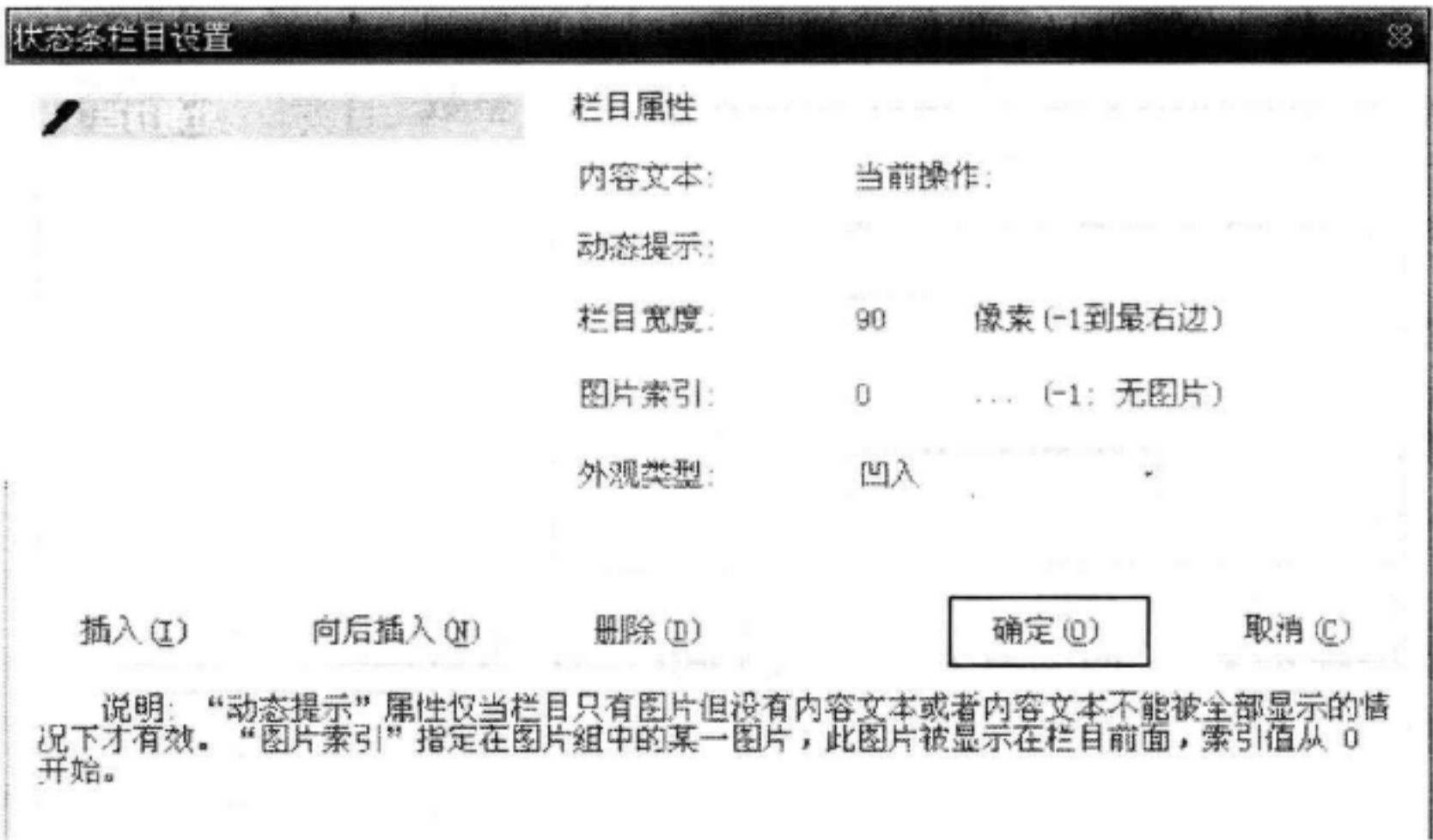


图 11-16 状态条栏目设置对话框

11.2.2 状态条组件的事件

状态条组件的事件有:鼠标左键被按下等。

“鼠标左键被按下”事件

此事件是一个常用事件,其经常性的用途是弹出一个菜单。程序代码如下:

子程序:_状态条 1_鼠标左键被按下

返回值类型:逻辑型

参数:横向位置 数据类型:整数型

参数:纵向位置 数据类型:整数型

参数:功能键状态 数据类型:整数型

如果真(横向位置 < 90)

 弹出菜单(弹出菜单, _启动窗口. 左边, _启动窗口. 顶边 + _启动窗口. 高度)

如果真结束

运行时在状态条的第一个栏目中单击会弹出菜单,在其他栏目单击不作动作。

11.2.3 状态条组件的方法

状态条组件的方法除去以“取”或“置”开头的命令,余下的命令与工具条组件的命令是一样的,如图 11-17 所示。

1. “清空()”方法

该方法会将状态条的内容全部清除掉。程序代码如下:

状态条 1. 清空()

2. “加入栏目()”方法

该方法会将指定栏目加入到状态条中,返回加入后该项目所处的位置。本命令为初级对象成员命令。其调用格式为:

〈整数型〉对象.加入栏目([文本型 栏目文本],[整数型 宽度],[文本型 动态提示文本],[整数型 图片索引],[整数型 外观类型])

参数“栏目文本”的类型为“文本型(text)”,可以省略。栏目文本指定栏目中的文本内容。如果本参数被省略,默认值为空文本。

参数“宽度”指定栏目的宽度,其类型为“整数型(int)”,可以省略。宽度单位为像素,如果值为-1,表示一直到窗口的最右边。如果被省略,默认值为50。

参数“动态提示文本”的类型为“文本型(text)”,可以省略。动态提示文本仅当栏目只有图片而没有内容文本或者内容文本不能全部显示的情况下才有效。如果本参数被省略,默认值为空文本。

参数“图片索引”的类型为“整数型(int)”,可以省略。图片索引用于指定图片组属性中的某张图片,该图片将被显示在栏目的首部。索引值从0开始,-1表示无图片。如果本参数被省略,默认值为-1。

参数“外观类型”的类型为“整数型(int)”,可以被省略。外观类型指定栏目的显示外观,可以为以下常量值之一:0. #凹入、1. #无边框、2. #凸出。如果本参数被省略,默认值为“0. #凹入”。

程序代码如下:

状态条 1. 加入栏目(“新栏目”,180,“这是加入的新栏目”,0,)

【注意】这个命令要在状态条清空的情况下才可以正常使用。

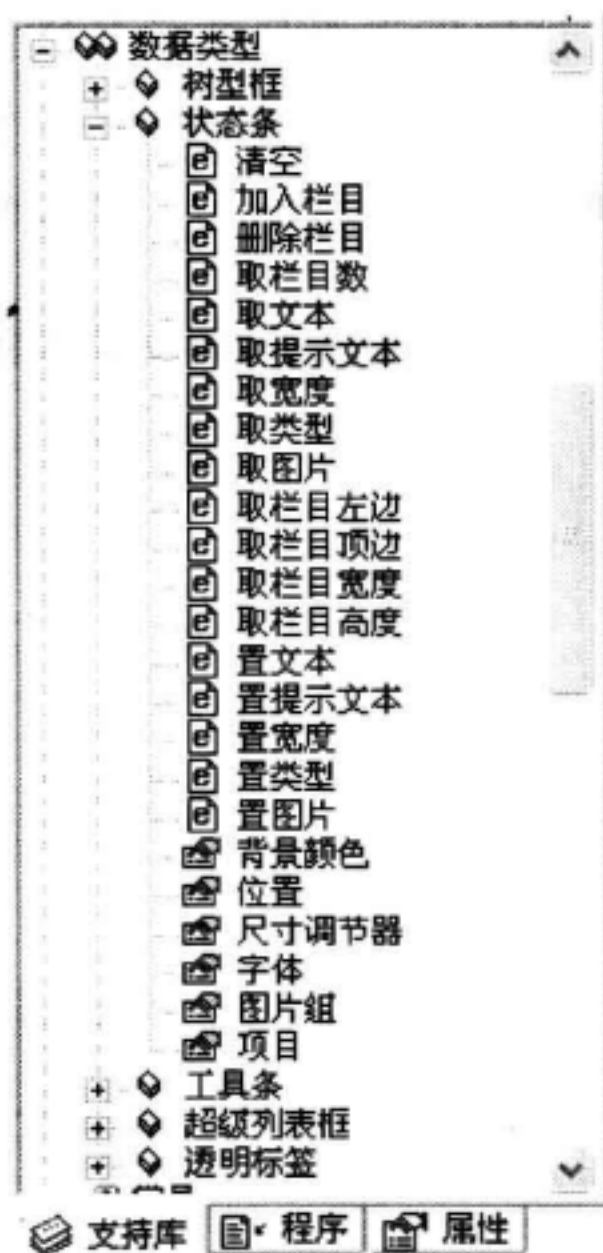


图 11-17 状态条组件的命令

3. “删除栏目()”方法

该方法可以将指定的栏目删除。程序代码如下：

状态条 1. 删除栏目(0)

4. “置文本()”方法

这是状态条组件最常用的组件命令之一,设置指定栏目的文本,用来显示不同的文本信息,其调用格式为:

对象. 置文本(整数型 栏目索引,[文本型 栏目文本])

参数“栏目索引”的类型为“整数型(int)”。其值 0 代表栏目一,1 代表栏目二,以此类推。

参数“栏目文本”的类型为“文本型(text)”,可以被省略。栏目文本指定栏目中的文本内容。如果本参数被省略,默认值为空文本。

相关程序代码如下:

状态条 1. 置文本(1,“使用置文本命令,加入提示信息!”)

此代码运行时会在第 2 个栏目中写入文本信息。

5. “取文本()”方法

该方法可以取到指定栏目的文本。程序代码如下:

信息框(“取到的文本是:” + 状态条 1. 取文本(1),0,)

6. “置宽度()”方法

状态条一般都是在设计时定好的,如果当提示信息的长度太长时就要重置宽度了,程序代码如下:

状态条 1. 置宽度(0,180)

此行代码将第一栏的宽度,重置为 180 像素。

7. “取宽度()”方法

此方法用来取到状态条或指定栏目的宽度,程序代码如下:

信息框(“取到的文本是:” + 到文本(状态条 1. 取宽度(1)),0,)

【范例 11-2】使用状态条组件的方法实现对状态条的控制。启动窗口中加入状态条和一系列按钮,如图 11-18 所示,点击相应按钮实现对状态条的操作。具体参考例程 11-2。

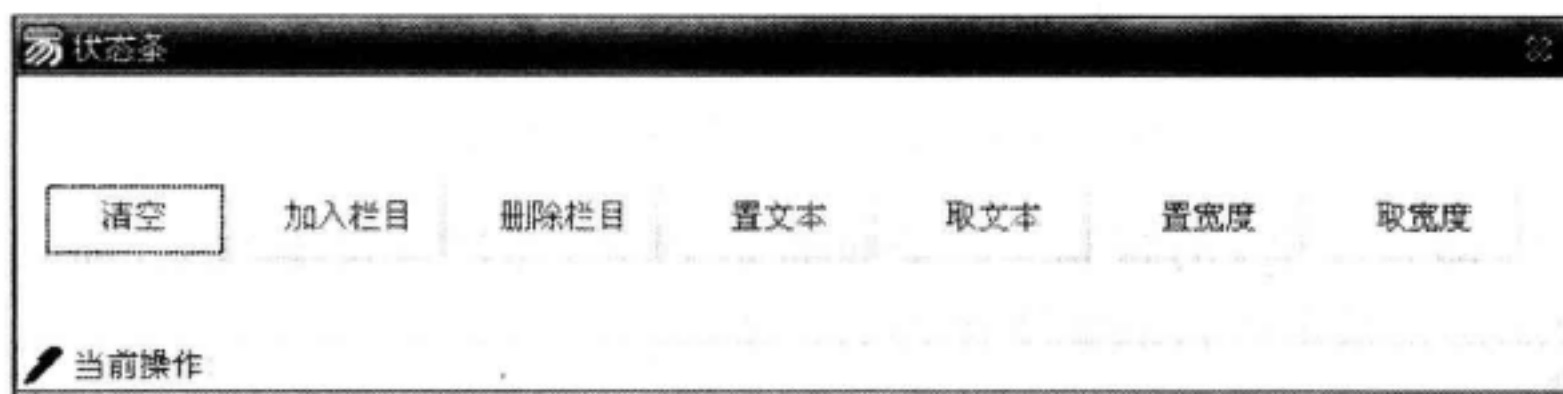


图 11-18 状态条组件例程设计窗口

启动窗口的程序代码如图 11-19 所示。

【运行结果】运行例程 11-2,观测单击不同按钮时状态条的变化,图 11-20 所示是先后点击了“清空”、“加入栏目”、“置文本”以及“取宽度”按钮后的效果。

【范例解析】“加入栏目”的功能是将标题为“新栏目”、宽度为 180 的状态条栏目添加到状态条中,可以多次点击添加。此按钮必须在状态条清空状态之后才有效;“置文本”的功能

窗口程序集名	保留	保留	备注
窗口程序集1			

子程序名	返回值类型	公开	备注
_按钮1_被单击			

| 状态条1.清空 (0) | | | |

子程序名	返回值类型	公开	备注
_按钮2_被单击			

| 状态条1.加入栏目 (“新栏目”, 180, “这是刚才加入的栏目”, 0,) | | | |

子程序名	返回值类型	公开	备注
_按钮3_被单击			

| 状态条1.删除栏目 (0) | | | |

子程序名	返回值类型	公开	备注
_按钮6_被单击			

| 状态条1.置文本 (1, “使用置文本命令,加入提示信息!”) | | | |

子程序名	返回值类型	公开	备注
_按钮4_被单击			

| 信息框 (“取到的文本是:” + 状态条1.取文本 (1), 0,) | | | |

子程序名	返回值类型	公开	备注
_按钮5_被单击			

| 状态条1.置宽度 (0, 180) | | | |

子程序名	返回值类型	公开	备注
_按钮7_被单击			

| 信息框 (“取到的宽度是:” + 到文本 (状态条1.取宽度 (0)), 0,) | | | |

图 11-19 启动窗口程序集

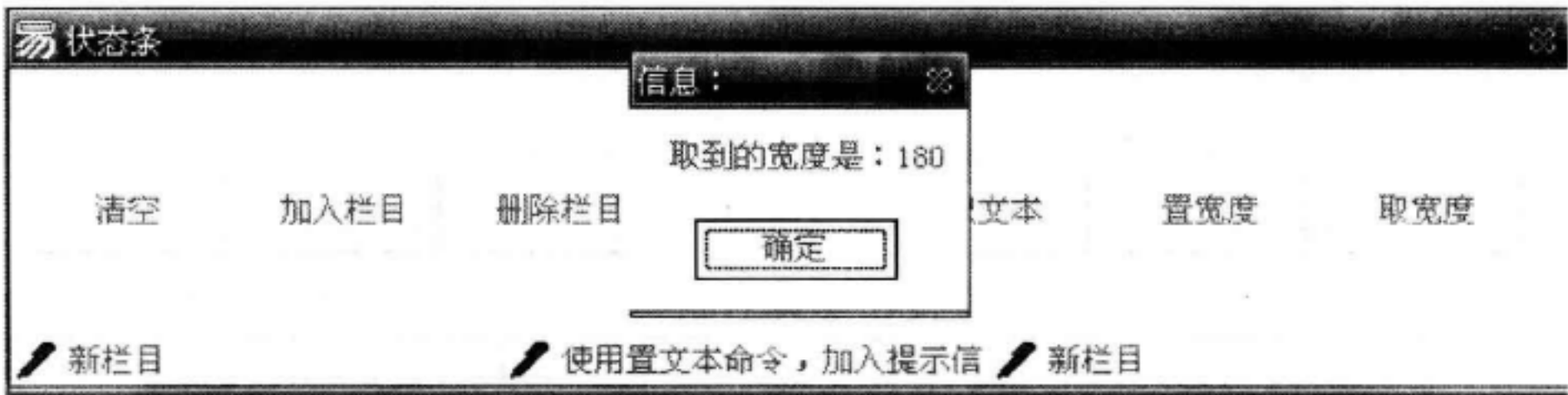



图 11-20 状态条组件例程运行结果

是将栏目号为 1(即第二个)的状态条栏目的标题置为“使用置文本命令,加入提示信息!”;“置宽度”的功能是将指定栏目的宽度置为 180 像素。

11.3 树形框组件

树形框组件 可以使其中的内容以层次的形式显示出来。

11.3.1 树形框组件的属性

树形框组件的属性如图 11-21 所示。

1. “显示加减框”属性

本属性为逻辑型,指定是否显示加减框,默认为“真”。图 11-22 所示为此属性为“真”和“假”的对比。

2. “显示连线”属性

本属性为逻辑型,指定是否显示连接线,默认为“真”。图 11-23 所示为此属性为“真”和“假”的对比。

树形框1 (树形框)

名称	树形框1
备注	
左边	152
顶边	152
宽度	16
高度	16
标记	
可视	真
禁止	假
鼠标指针	默认型
可停留焦点	真
停留顺序	0
边框	凹入式
文本颜色	黑色
背景颜色	白色
显示加减框	真
显示连线	真
显示根部线	假
允许编辑	假
始终显示选择项	假
现行选中项	-1
行高	-1
字体	
图片组	
项目	
结束编辑文本	** 只读属性 **
是否有检查框	假
检查框图片组	
检查框状态数	2

图 11-21 树形框组件的属性



图 11-22 “显示加减框”属性对比



图 11-23 “显示连线”属性对比

3. “显示根部线”属性

本属性为逻辑型,指定是否从根部显示连接线,默认为“假”。图 11-24 所示为此属性为“真”和“假”的对比。

4. “允许编辑”属性

本属性为逻辑型,指定是否允许编辑标签内容。默认为“假”。

本属性设置为“真”后运行,先激活要编辑的一行项目,再单击一次,就会生成编辑状态,如图 11-25 所示。

【注意】使用命令也可以让选中项处于编辑状态。使用的程序代码为:

树形框 1. 进入编辑 (树型框 1. 现行选中项)

5. “始终显示选择项”属性

本属性为逻辑型,指定是否始终显示当前被选择项目,即使树形框失去焦点。默认为“假”。

图 11-26 所示,第 1 个树型框的“始终显示选择项”属性为“真”,当焦点不存在第 1 个树形框时,即默认选择或刚才所选择的项目为灰色。而第 2 个树型框的始终显示选择项属性为“假”,所以当焦点移开时,即没有任何变色项目指示。

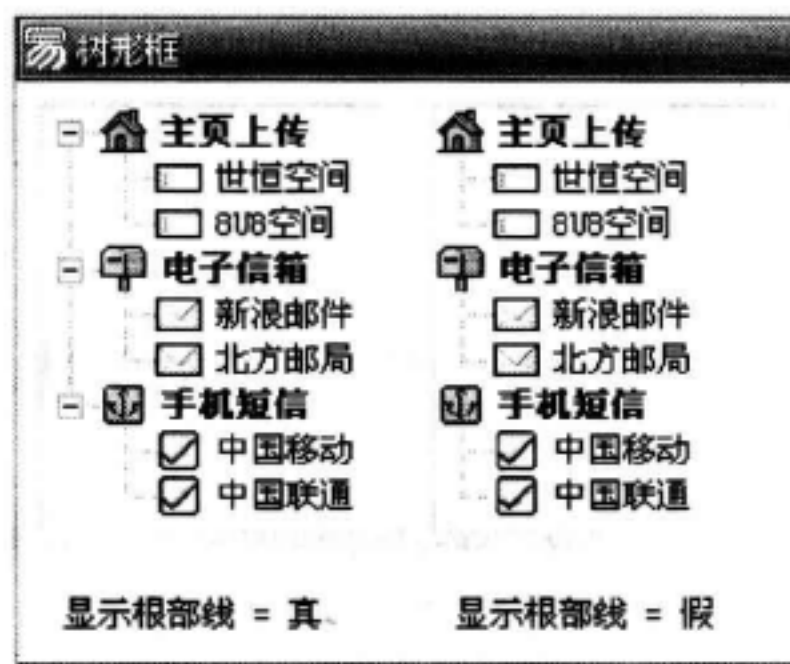


图 11-24 “显示根部线”属性对比

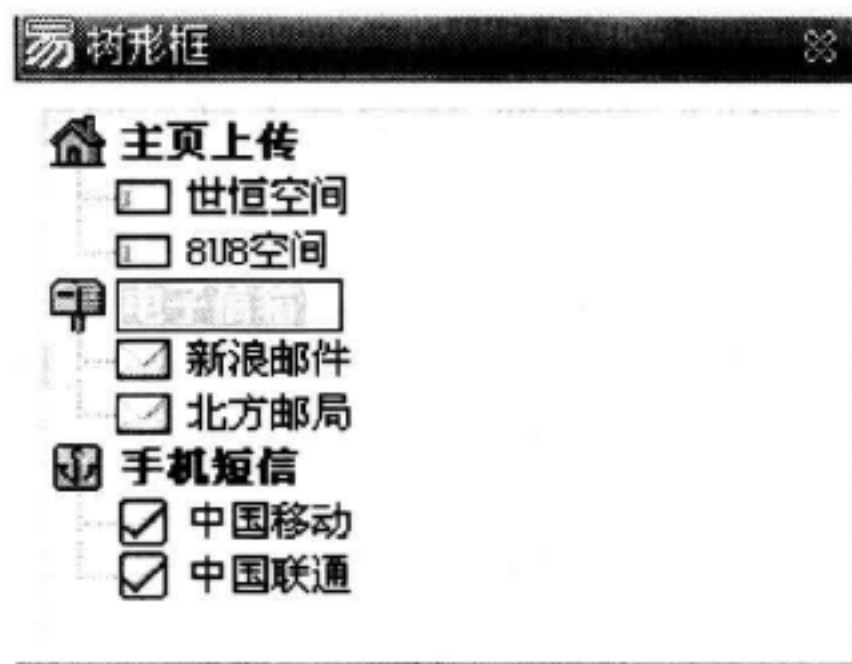


图 11-25 “允许编辑”属性对比



图 11-26 “始终显示选择项”属性对比

6. “现行选中项”属性

本属性是使用最多的一个属性,指定树形框中现行被选中项目的位置,位置值从 0 开始,值为 -1 则表示现行没有被选中的项目。程序代码如下:


编辑框 1. 内容 = 到文本(树形框 1. 现行选中项)

此语句通过项目的改变,就会将当前项目的索引显示在编辑框中。

7. “图片组”属性

此属性为树形框提供标志图片,其操作方法与工具条组件及状态条组件的载入图片方法一样。

8. “项目”属性

本属性指定各树型框项目。单击本属性后面的按钮,可以进入树形框项目设置对话框,如图 11-27 所示。

在弹出的对话框中,可以对每个行项目的标题、图片索引、选中后图片索引及项目数值等属性进行设置。

11.3.2 树形框组件的事件

树形框组件的事件有:项目被选择、双击项目、开始编辑、结束编辑、鼠标右键被按下、即将扩展、即将收缩、检查框状态被改变。

1. “项目被选择”事件

当前选择项目被改变时即产生此事件。如以下程序所示:

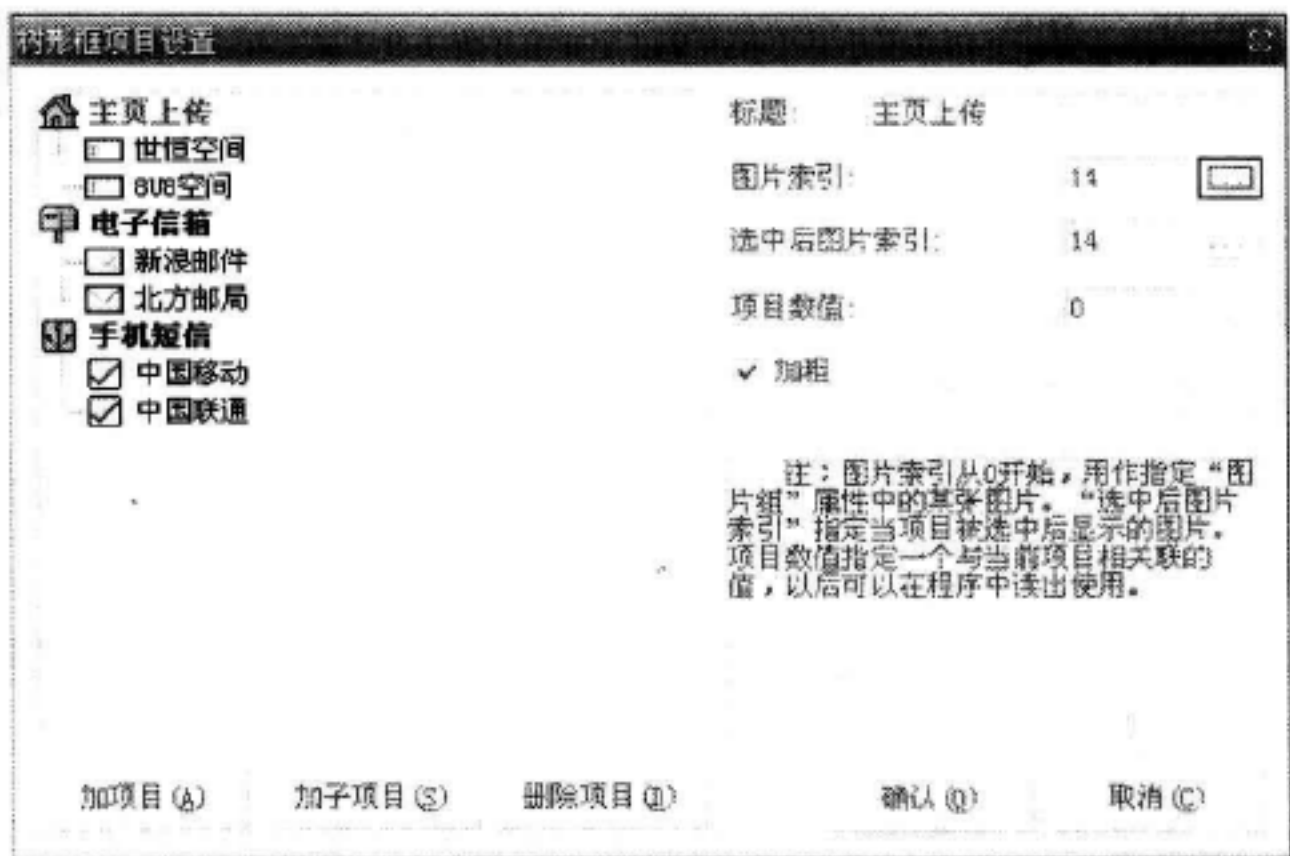


图 11-27 树形框项目设置对话框

子程序:_树形框 1_项目被选择
参数:选择方式 数据类型:整数型
编辑框 1. 内容 = 到文本(树形框 1. 现行选中项)

2. “开始编辑”事件

当项目进入标题编辑状态前产生此事件,要产生本事件“允许编辑”属性必须为“真”。事件处理子程序返回“假”则不允许进入编辑状态,返回“真”或不返回值允许进入。

3. “结束编辑”事件

当项目结束标题编辑状态前产生此事件,要产生本事件“允许编辑”属性必须为“真”。在事件处理子程序中读取“结束编辑文本”属性可以得知标题被编辑后的内容,事件处理子程序返回“假”不允许使用编辑后的内容替换原有标题,返回“真”或不返回值则允许替换。

4. “双击项目”事件

当使用鼠标左键在某项目上双击时产生此事件。

5. “被双击”事件

在树形框的空白处双击,就会产生本事件。处理本事件的子程序如果返回“假”,则取消本事件;如果不返回值或返回“真”,则将此事件继续传递到所属对象上去。

6. “鼠标右键被按下”事件

当使用鼠标右键单击时产生本事件。处理本事件的子程序如果返回“假”,则取消本事件,不再将此事件传递到所属对象上去;如果不返回值或返回“真”,则将此事件继续传递到所属对象上去。

11.3.3 树形框组件的方法

树形框组件现有 21 个专有方法,可以在支持库面板的“扩展界面支持库一”中找到,如图 11-28 所示。

1. “取项目数()”方法

本方法用于返回树形框中所有项目的数目(整数型),即

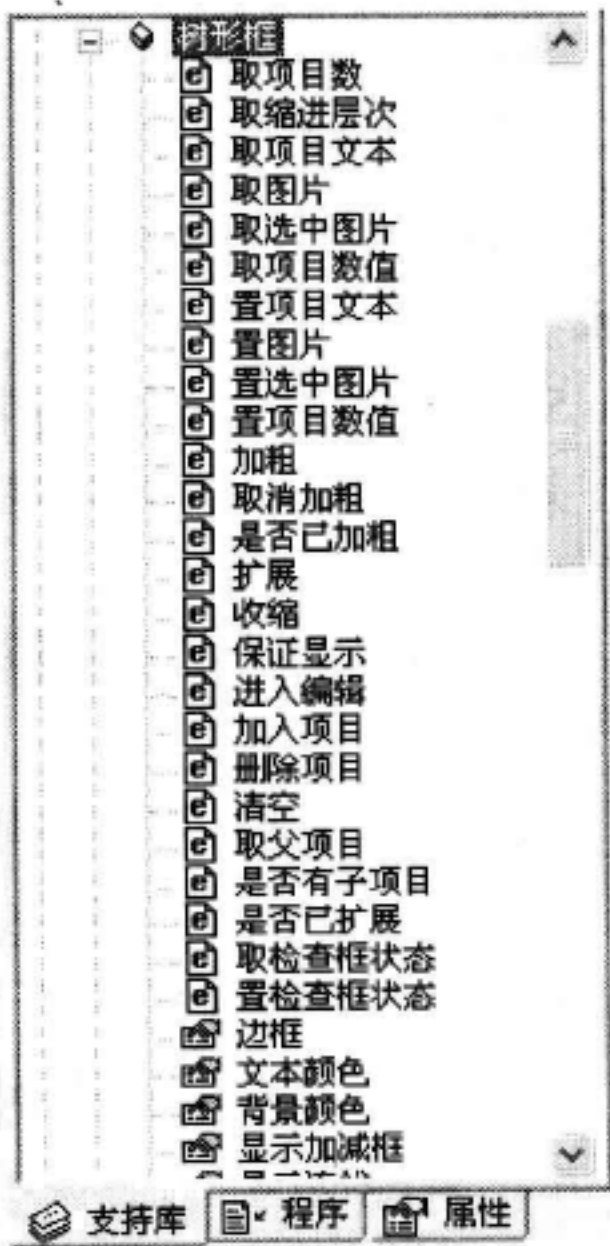


图 11-28 树形框的方法

所有项目的总数。程序代码如下所示：

编辑框 1. 内容 = “当前所有项目总数为:” + 到文本(树形框 1. 取项目数())

2. “取缩进层次()”方法

本命令为初级对象成员命令,用于返回指定项目所处缩进层次,层次值为项目索引,从 1 开始,如果未找到指定项目,则返回 0。程序代码如下:

编辑框 1. 内容 = “当前缩进层次为:” + 到文本(树形框 1. 取缩进层次(树形框 1. 现行选中项))

3. “取项目文本()”方法

本命令为初级对象成员命令,返回指定项目的文本,如果该项目不存在,将返回空文本。程序代码如下所示:

编辑框 1. 内容 = “当前项目文本为:” + 到文本(树形框 1. 取项目文本(树形框 1. 现行选中项))

此语句将当前选中的项目文本显示在编辑框中。

4. “置项目文本()”方法

本命令为初级对象成员命令,设置指定项目的文本,成功返回“真”,失败则返回“假”。其调用格式为:

<逻辑型> 对象. 置项目文本(项目索引,欲置入的项目文本)

参数“项目索引”的类型为“整数型(int)”。索引号从 0 开始,即 0 为项目一,1 为项目二,以此类推。

参数“欲置入的项目文本”的类型为“文本型(text)”。

相关程序代码如下所示:

子程序:_按钮 1_被单击

局部变量:文本变量 数据类型:文本型

输入框(“请输入项目文本”,,,文本变量,)

树形框 1. 置项目文本(树形框 1. 现行选中项,文本变量)

树形框 1. 刷新显示()

上述代码实现了弹出一个输入框,输入文本后,所选中的项目的文本被更换为输入的文本。

5. “取图片()”、“取选中图片()”方法

处于选中状态与非选中状态的小图标图片可以不一样,可以在树形框项目设置对话框中进行设置。

“取图片()”方法返回指定项目的图片索引(数值型)。如果该项目不存在,将返回 -1。程序代码如下:

编辑框 1. 内容 = “当前图片索引为:” + 到文本(树形框 1. 取图片(树形框 1. 现行选中项))

“取选中图片()”方法返回指定项目被选中后的图片索引(数值型)。如果该项目不存在,将返回 -1。程序代码如下:

编辑框 1. 内容 = “当前选中图片索引为:” + 到文本(树形框 1. 取选中图片(树形框 1. 现行选中项))

6. “置图片()”、“置选中图片()”方法

(1) “置图片()”方法设置指定项目首部所显示图片的索引(数值型),成功返回“真”,失败则返回“假”。其调用格式为:

<逻辑型> 对象. 置图片(项目索引, 图片索引)

参数“项目索引”的类型为“整数型(int)”。索引号从0开始, 0为项目一, 1为项目二, 以此类推。

参数“图片索引”的类型为“整数型(int)”。图片索引用于指定图片组属性中的某张图片, 索引号从0开始。

程序代码如下所示:

```
树形框 1. 置图片(树形框 1. 现行选中项, 变量 1)
树形框 1. 刷新显示()
```

(2) “置选中图片()”方法设置指定项目被选中后所显示图片的索引(数值型)。成功返回“真”, 失败返回“假”。其调用格式为:

<逻辑型> 对象. 置选中图片(项目索引, 图片索引)

参数“项目索引”的类型为“整数型(int)”。索引号从0开始, 0为项目一, 1为项目二, 以此类推。

参数“图片索引”的类型为“整数型(int)”。图片索引用于指定图片组属性中的某张图片, 索引号从0开始。

程序代码如下所示:

```
树形框 1. 置选中图片(树形框 1. 现行选中项, 变量 1)
树形框 1. 刷新显示()
```

7. “取项目数值()”方法

本方法返回与指定项目相关联的数值。如果该项目不存在, 将返回0。

程序代码如下:

```
编辑框 1. 内容 = “当前项目数值为:” + 到文本(树形框 1. 取项目数值(树形框 1. 现行选中项))
```

【注意】初始项目数值也可以通过树形框项目设置对话框进行设置。

8. “置项目数值()”方法

本方法设置与指定项目相关联的数值。成功返回“真”, 失败返回“假”。其调用格式为:

<逻辑型> 对象. 置项目数值(项目索引, 欲置入的项目数值)

参数“项目索引”的类型为“整数型(int)”。索引号从0开始, 0为项目一, 1为项目二, 以此类推。

参数“欲置入的项目数值”的类型为“整数型(int)”, 可以被省略, 该数值与指定项目相关联。如果本参数被省略, 默认值为0。程序代码如下:

```
树形框 1. 置项目数值(树形框 1. 现行选中项, 变量 1)
树形框 1. 刷新显示()
```

9. “加粗()”、“取消加粗()”方法

“加粗()”方法可以将指定项目的标题加粗显示, “取消加粗()”方法将指定项目的标题取消加粗显示。程序代码如下:

```
树形框 1. 加粗(树形框 1. 现行选中项)
树形框 1. 取消加粗(树形框 1. 现行选中项)
```

10. “扩展()”、“收缩()”方法

“扩展()”方法用于打开指定项目的下属分支,“收缩()”方法缩回指定项目的下属分支。程序代码如下:

```
树形框 1. 扩展(树形框 1. 现行选中项)
```

```
树形框 1. 收缩(树形框 1. 现行选中项)
```

11. “进入编辑()”方法

本方法可以让指定项目进入编辑状态,执行本命令成功的前提是“允许编辑”属性为真。程序代码如下:

```
树形框 1. 进入编辑(树形框 1. 现行选中项)
```

12. “加入项目()”方法

本方法可以将指定项目加入到树形框中,若成功则返回加入后该项目所处的位置,失败则返回 -1。其调用格式为:

<整数型> 对象.加入项目([父项目索引],项目文本,[图片索引],[选中图片索引],[项目数值],[是否加粗],[检查框状态])

(1) 参数“父项目索引”的类型为“整数型(int)”,可以被省略。指定欲加入项目所处的父项目,0 为项目一,1 为项目二,如此类推。如果没有父项目(即欲加入项目为顶层项目),请提供值 -1。如果本参数被省略,默认值为 -1。

(2) 参数“项目文本”的类型为“文本型(text)”。

(3) 参数“图片索引”的类型为“整数型(int)”,可以被省略。图片索引用于指定图片组属性中的某张图片,从 0 开始。如果本参数被省略,默认值为 0。

(4) 参数“选中图片索引”的类型为“整数型(int)”,可以被省略。选中图片索引用于指定项目被选中后所显示的图片,-1 表示与图片索引一致。如果本参数被省略,默认值为 -1。

(5) 参数“项目数值”的类型为“整数型(int)”,可以被省略。指定与本项目相关联的数值。如果本参数被省略,默认值为 0。

实际操作中,可以在同级项目中增加一个新项目,也可以在项目中增加一个子项目,相关程序代码如下:

```
子程序:_加入项目_被选择
```

```
局部变量:变量 1 数据类型:文本型
```

```
输入框(“请输入一个数值”,,,变量 1,)
```

```
树形框 1. 加入项目(树形框 1. 取父项目(树形框 1. 现行选中项),变量 1,,)
```

```
树形框 1. 刷新显示()
```

```
子程序:_加入子项目_被选择
```

```
局部变量:变量 1 数据类型:文本型
```

```
输入框(“请输入一个数值”,,,变量 1,)
```

```
树形框 1. 加入项目(树形框 1. 现行选中项,变量 1,,)
```

```
树形框 1. 刷新显示()
```

13. “删除项目()”方法

本方法为逻辑型,用于删除树形框中的指定项目。成功返回“真”,失败返回“假”。程序

代码如下：

树形框 1. 删除项目(树形框 1. 现行选中项)

14. “清空()”方法

此方法可以删除树形框中的所有项目。程序代码如下：

树形框 1. 清空()

【范例 11 - 3】树形框中用代码实现添加删除项目或子项目。设计窗口如图 11 - 29 所示。使用菜单编辑器编辑本例中的右键弹出子菜单如图 11 - 30 所示。具体参考例程 11 - 3。



图 11 - 29 树形框例程设计窗口

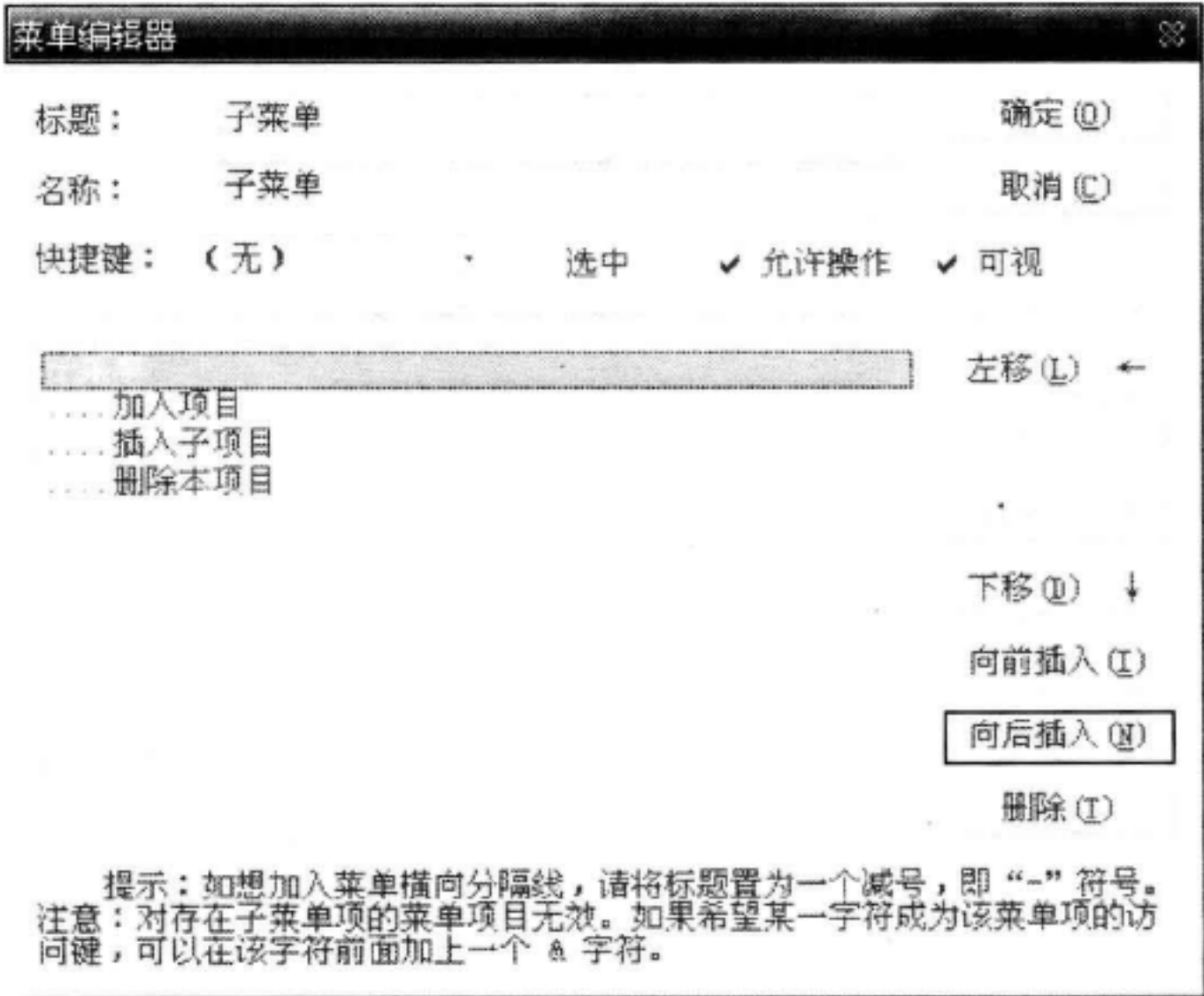


图 11 - 30 子菜单编辑窗口

启动窗口的程序代码如图 11 - 31 所示。

【运行结果】运行例程 11 - 3,观测利用右键弹出的菜单任意添加或删除树形框中的项目和子项目的效果,如图 11 - 32 所示。

【范例解析】右键弹出菜单设计了三个项目:插入子项目、加入项目和删除本项目。选择“加入项目”将在现行选中项的同级创建一个新项目,选择“插入子项目”则在现行选中项的下级创建一个项目,选择“删除本项目”则删除现行选中项。

窗口程序集名	保 留	保 留	备 注
窗口程序集1			
变量名	类 型	数 组	备 注
项目标题	文本型		

子程序名	返回值类型	公开	备 注
__启动窗口_创建完毕			

树形框1.加入项目 (0, “今天”, , ,)

树形框1.加入项目 (1, “早晨”, , , , ,)

子程序名	返回值类型	公开	备 注		
_树形框1_鼠标右键被按下	逻辑型				
参数名	类 型	参 考	可 空	数 组	备 注
横向位置	整数型				
纵向位置	整数型				
功能键状态	整数型				

弹出菜单 (子菜单, ,)

子程序名	返回值类型	公开	备 注
_插入子项目_被选择			

输入框 (“请输入子项目标题”, , , 项目标题,)

树形框1.加入项目 (树形框1.现行选中项, 项目标题, , ,)

树形框1.扩展 (树形框1.现行选中项)

子程序名	返回值类型	公开	备 注
_删除本项目_被选择			

树形框1.删除项目 (树形框1.现行选中项)

子程序名	返回值类型	公开	备 注
_加入项目_被选择			

输入框 (“请输入项目标题”, , , 项目标题,)

树形框1.加入项目 (树形框1.取父项目 (树形框1.现行选中项), 项目标题, , ,)


树形框1.扩展 (树形框1.现行选中项)

图 11-31 启动窗口程序集



图 11-32 树形框例程运行结果

11.4 超级列表框组件

超级列表框组件  可以使其中的内容以列表的形式表现出来。

11.4.1 超级列表框组件的属性

超级列表框组件的重要属性如图 11-33 所示。

1. “类型”属性

本属性指定列表框的类型。可从 4 种类型中选择:0. 大图标列表框、1. 小图标列表框、2. 普通列表框、3. 报表列表框。这 4 种列表框有各自的形态,类似于操作系统的浏览器,可以设定查看方式为大图标、小图标、报表或详细资料等。

2. “标题自动换行”属性

本属性指定在大图标和小图标列表框中如果标题过长是否自动换行。一般将本属性设置为“真”。

3. “报表列”属性

本属性设置报表列表框中各列的信息。

【注意】报表列表框只有在设置了列后才能显示出其中的内容。

“报表列”属性只有在“类型”属性为“3. 报表列表框”时才允许操作。其他同时被允许操作的还有:“无表头”、“表头图片组”、“表头可单击”、“整行选择”、“显示表格线”、“表列可拖动”几个属性。而同时“图标对齐方式”、“自动排列图标”、“标题自动换行”几个属性不可操作。

4. “无表头”属性

当“无表头”属性设置为“真”时,将不显示“报表列”属性中所设置的表头,反之显示。此属性设为“假”时列表框的显示效果如图 11-34 所示。

5. “表头图片组”属性

本属性为报表列表框表头提供图标图片。表头图片组与图片组的操作方式一样。

6. “显示表格线”属性

本属性指定在报表列表框中是否显示表格线。

当“显示表格线”为“真”时,行列间隙处有表格线出现,如图 11-34 所示。反之当“显示表格线”为“假”时,行列间隙处没有表格线出现,如图 11-35 所示。

超级列表框1 (超级列表框)	
高度	40
标记	
可视	真
禁止	假
鼠标指针	默认型
可停留焦点	真
停留顺序	0
边框	凹入式
文本颜色	黑色
文本背景色	白色
背景颜色	白色
字体	
类型	大图标列表框
图标对齐方式	顶部对齐
自动排列图标	真
标题自动换行	真
报表列	
无表头	假
表头图片组	
表头可单击	真
整行选择	假
显示表格线	假
表列可拖动	假
图片组	
状态图片组	
热点跟踪	假
手形指针	真
标注非热点	真
标注热点	真
允许编辑	假
排序方式	不排序
是否有检查框	假
平面滚动条	假
单一选择	真
现行选中项	-1
始终显示选择项	假
表项	
结束编辑文本	** 只读属性 **

图 11-33 超级列表框的属性

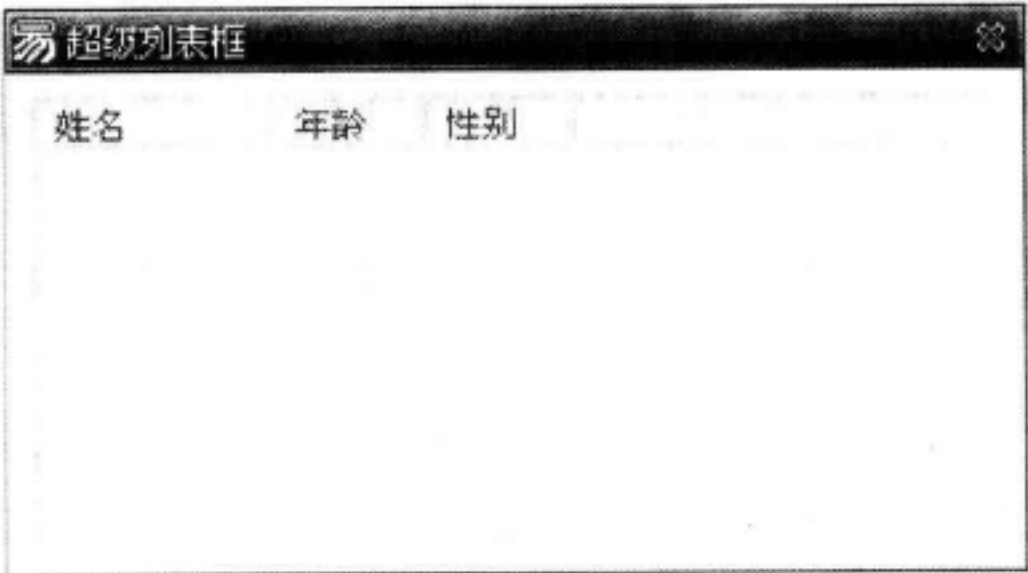


图 11-34 “无表头”属性设置为“假”

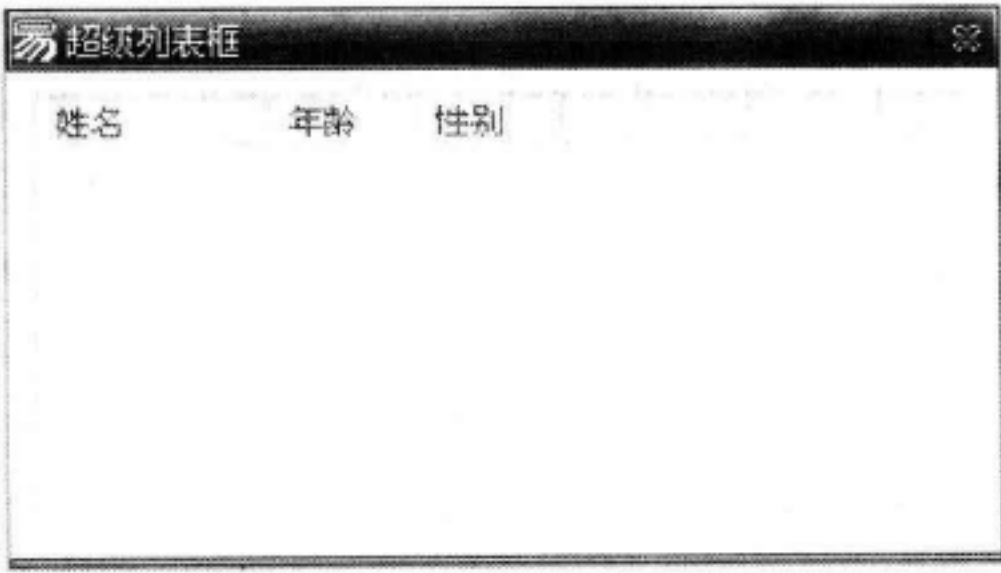



图 11-35 “显示表格线”属性设置为“假”

7. “图片组”属性

本属性为各列表框项目提供图标图片。与其他组件一样,图片组的操作基本是相同的,点

击此属性后的按钮,进入图片组设置对话框进行设置。

8. “热点跟踪”属性

本属性指定在列表框中是否跟踪并高亮度显示当前鼠标所处表项,此表项称为“热点表项”。若超级列表框中的“热点跟踪”属性设为“真”,则运行时,将鼠标放在组件中,可以看到鼠标下方的对应激活色块,鼠标变为手形,单击即可选中。

9. “允许编辑”属性

本属性指定是否允许编辑表项标题,仅在“热点跟踪”属性为“假”时才有效。

【注意】在运行时单击两次,就会进入编辑状态,而不必双击或用鼠标右击。

10. “排序方式”属性

本属性指定列表框表项的标题排序方式。有3个可选项:0. 不排序、1. 正向排序、2. 逆向排序。

【注意】本属性的排序只针对于超级列表框的第一栏。

11. “状态图片组”属性

本属性为列表框中的表项提供状态图片。一般用于在前面加勾、检查框等符号之用。与“是否有检查框”属性有直接的联系。

12. “是否有检查框”属性

本属性指定是否在表项前加上检查框,如果“状态图片组”属性被设置了相关内容,则默认的检查框图片将被“状态图片组”中的图片代替,并且当用户单击表项中的状态图片时,将自动遍历显示“状态图片组”中的所有图片。下面对“状态图片组”和“是否有检查框”属性举一个例子,如图11-36所示。

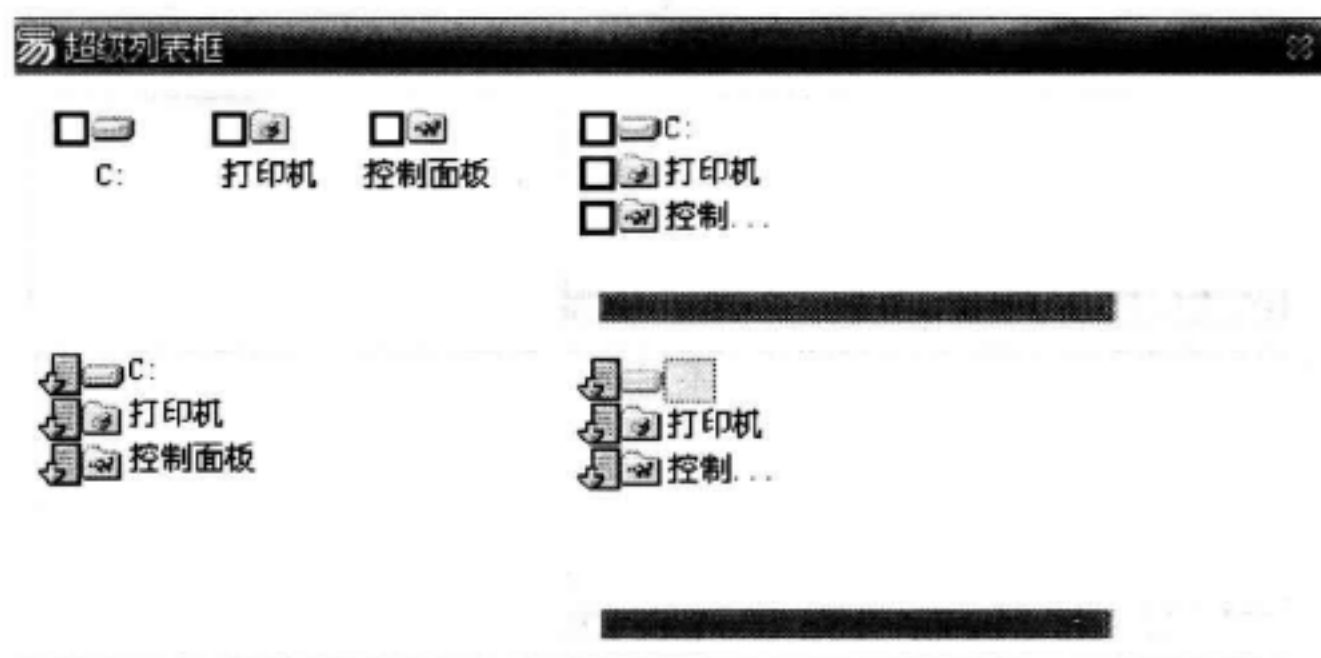


图 11-36 超级列表框-“状态图片组”及“是否有检查框”属性

第1个超级列表框“类型”属性为“大图标列表框”,“状态图片组”属性为空,“是否有检查框”属性为“真”,因此运行时会显示一排选择框。

第2个超级列表框“类型”属性为“报表列表框”,“状态图片组”属性为空,“是否有检查框”属性为“真”,因此运行时会显示一排选择框。

第3个超级列表框“类型”属性为“小图标列表框”,“状态图片组”属性为有数据,“是否有检查框”属性为“真”,因此运行时单击项目,会将状态图片组中的图标顺序显示出来。

第4个超级列表框“类型”属性为“报表列表框”,“状态图片组”属性为有数据,“是否有检查框”属性为“真”,因此运行时单击项目,会将状态图片组中的图标顺序显示出来。

13. “平面滚动条”属性

本属性指定是否以平面方式显示列表框滚动条。

14. “单一选择”属性

本属性指定列表框中是否同时只允许选择一个表项。如果“单一选择”为“假”，且“类型”属性为非报表列表框时，即可以通过鼠标拖动，同时选择多个。

【注意】设置“单一选择”属性为“假”后，已不能使用“现行选中项”这个属性进行判断，如要判断，它也只能显示第 1 个所选择的项目，因此要使用其他的属性项目是否被选中进行判断。

15. “现行选中项”属性

本属性指定列表框中现行被选中项目的位置，位置值从 0 开始，若值为 -1 表示现行没有被选中的项目。


【注意】如果“单一选择”属性为“假”，则本属性只可读。如果“单一选择”属性为“真”，此属性即提示默认的选择项。

此属性是超级列表框使用最多的一个属性，因为所有用鼠标单击超级列表框后都要由本属性取得单击的是哪一个项目。

16. “始终显示选择项”属性

本属性指定是否始终显示当前被选择项目，即使列表框失去焦点，也有灰色块表示选中状态。

17. “表项”属性

本属性指定列表框中的各表项。单击本属性后面的按钮，即弹出超级列表框表项设置对话框，如图 11-37 所示。

本属性的设置与工具条组件中的“按钮”属性或状态条组件中的“项目”属性相似。

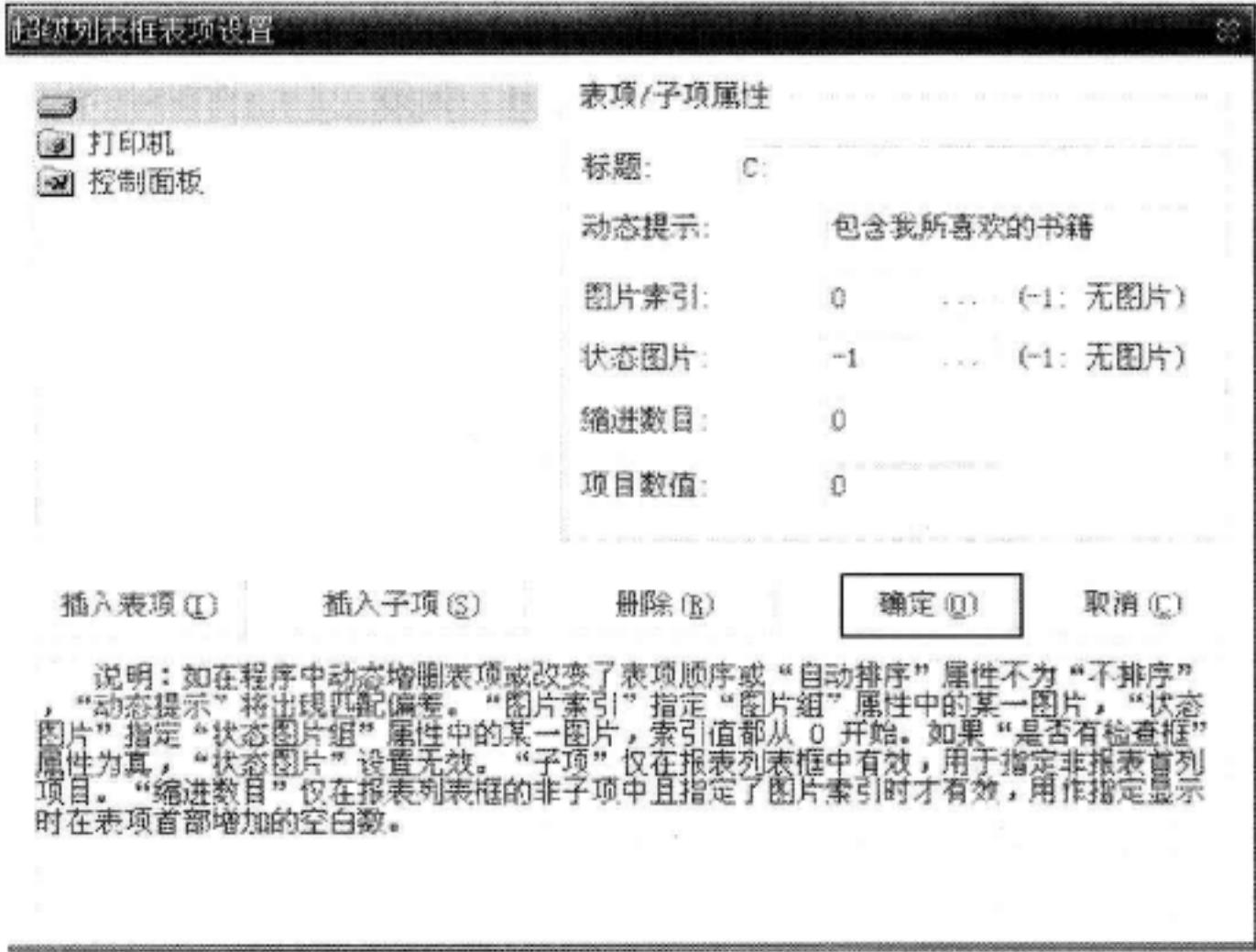


图 11-37 超级列表框表项设置对话框

同样的功能也可以通过程序代码实现，程序代码如下所示：

```
超级列表框 1. 置标题(0,,"C:")
超级列表框 1. 置图片(0,,1)
超级列表框 1. 置标题(1,,"打印机")
超级列表框 1. 置图片(1,,8)
超级列表框 1. 置标题(2,,"控制面板")
超级列表框 1. 置图片(2,,6)
```

11.4.2 超级列表框组件的事件

超级列表框组件的事件有：当前表项被改变、表项被激活、表头被单击、表项跟踪、左键单击表项、右键单击表项、开始编辑、结束编辑及检查框状态被改变。

1. “当前表项被改变”事件

如果当前焦点所处表项被改变即产生此事件，接收到此事件后读取“现行选中项”属性即可获得当前表项的索引。

2. “表项被激活”事件

在下列情况下会产生本事件：

- (1) 双击某表项；
- (2) 选中某表项后按回车；
- (3) 当“热点跟踪”属性为“真”时单击某表项。

接收到此事件后读取“现行选中项”属性即可获得当前被激活表项的索引。

3. “表头被单击”事件

当使用鼠标左键在报表型列表框的某表头列上单击时产生此事件。产生的事件子程序中有系统参数“被单击列索引”，可以根据这个参数知道是单击了表头的哪一列，从而进行相应的操作。

4. “表项跟踪”事件

当“热点跟踪”属性为“真”时移动鼠标到某表项上后即产生本事件，如果鼠标不处于任何表项上，“表项索引”参数将传递值 -1。如图 11-38 所示，运行中，并没有进行任何的单击，只不过鼠标移动到有内容的项目上，即产生本事件。

5. “左键单击表项”事件

当用鼠标左键单击某表项时即产生此事件，接收到此事件后读取“现行选中项”属性即可获得被单击表项的索引。

6. “右键单击表项”事件

当用鼠标右键在某表项上单击时产生此事件，接收到此事件后读取“现行选中项”属性即可获得被单击表项的索引，调用系统核心支持库中的“取鼠标水平位置()”、“取鼠标垂直位置()”命令可获取当前鼠标单击位置。



图 11-38 超级列表框事件

11.4.3 超级列表框组件的方法

本组件的专有方法非常多，有 35 个方法。部分的方法与树形框中的方法基本一样，因此此处只介绍部分常用方法的使用。

1. “全部删除()”方法

该方法用于删除所有的表项。相关的程序命令代码为：

超级列表框 1. 全部删除()

2. “插入表项()”方法

该方法在指定的位置插入新表项，成功则返回新表项的位置索引，失败则返回 -1。本命

令为初级对象成员命令。其格式为：

对象.插入表项([整数型 插入位置],[文本型 标题],[整数型 图片索引],[整数型 状态图片索引],[整数型 缩进数目],[整数型 表项数值])

参数“插入位置”的类型为“整数型(int)”,可以被省略。指定新表项插入时的位置索引,索引值从 0 开始。如果提供 -1,则插入到列表框的尾部。如果本参数被省略,默认值为 -1。

参数“标题”的类型为“文本型(text)”,可以被省略。本参数指定表项的标题文本。如果被省略,默认值为空文本。

参数“图片索引”的类型为“整数型(int)”,可以被省略。图片索引用于指定“图片组”属性中的某张图片。索引值从 0 开始,-1 表示无图片。如果本参数被省略,默认值为 -1。

参数“状态图片索引”的类型为“整数型(int)”,可以被省略。状态图片索引用于指定“状态图片组”属性中的某张图片。索引值从 0 开始,-1 表示无图片。如果本参数被省略,默认值为 -1。

参数“缩进数目”的类型为“整数型(int)”,可以被省略。“缩进数目”仅在报表型列表框中才有效,用作指定显示时在表项首部增加的空白数。如果本参数被省略,默认值为 0。

参数“表项数值”的类型为“整数型(int)”,可以被省略。该数值与指定表项相关联。如果本参数被省略,默认值为 0。

相关的程序命令代码如下所示：

```
变量 1 = 超级列表框 1. 插入表项(,“易语言”,,,)
```

3. “置标题()”方法

该方法设置指定表项或子项的标题。其基本格式为：

对象.置标题(整数型 表项索引,[整数型 列索引],[文本型 标题])

参数“表项索引”的类型为“整数型(int)”。用于指定列表框中的某一项目,索引值从 0 开始。

参数“列索引”的类型为“整数型(int)”,可以被省略。用作指定报表型列表框中项目所处的列,索引值从 0 开始,如果当前列表框类型不是报表型列表框,提供 0 值即可。如果本参数被省略,默认值为 0。

参数“标题”的类型为“文本型(text)”,可以被省略。本参数指定表项或子项的标题文本。如果被省略,默认值为空文本。

相关的程序命令代码如下：

```
超级列表框 1. 置标题(变量 1,1,“易语言”)
```

4. “取表项数()”方法

返回列表框中所有表项的数目。这里是取表的横排的数值,如果想取列排,请使用“取列数()”命令。

5. “取当前状态图片()”方法

该方法可以取回指定表项的当前状态图片的索引,图片索引指向“状态图片组”属性中的某一图片,从 0 开始,-1 表示无图片。如果“是否有检查框”属性为“真”,返回值是当前检查框的选中状态,1 表示被选中,0 表示未选中。本命令为初级对象成员命令。

该方法的参数名称为“表项索引”,类型为“整数型(int)”。用于指定列表框中的某一项目,索引值从 0 开始。相关程序代码为：

信息框(“超级列表框 1 第 2 项是否被选择 =” + 到文本(超级列表框 1. 取当前状态图片(1)),0,)

如果状态图片没有数据,状态值为 0 或 1,如果状态图片有数据,就会根据选中的小图标图片的索引显示数值。

6. “取标题()”方法

该方法用于取指定表项或子项的标题。本命令为初级对象成员命令。其格式为:

对象.取标题(整数型 表项索引,[整数型 列索引])

参数“表项索引”的类型为“整数型(int)”。用于指定列表框中的某一项目,索引值从 0 开始。

参数“列索引”的类型为“整数型(int)”,可以被省略。用作指定报表型列表框中项目所处的列,索引值从 0 开始,如果当前列表框类型不是报表型列表框,提供 0 值即可。如果本参数被省略,默认值为 0。

7. “是否被选择()”方法

该方法判断指定表项是否已经被选择,如已被选择,返回“真”,否则返回“假”。本命令为初级对象成员命令。

该方法的参数名称为“表项索引”,类型为“整数型(int)”。用于指定列表框中的某一项目,索引值从 0 开始。

【范例 11-4】超级列表框表项信息的添加删除和编辑。启动窗口中添加一个超级列表框,设置其“报表列”属性(如图 11-39 所示)和“表项”属性(如图 11-40 所示),同时设计一个弹出菜单如图 11-41 所示,启动窗口的设计效果如图 11-42 所示。具体参考例程 11-4。

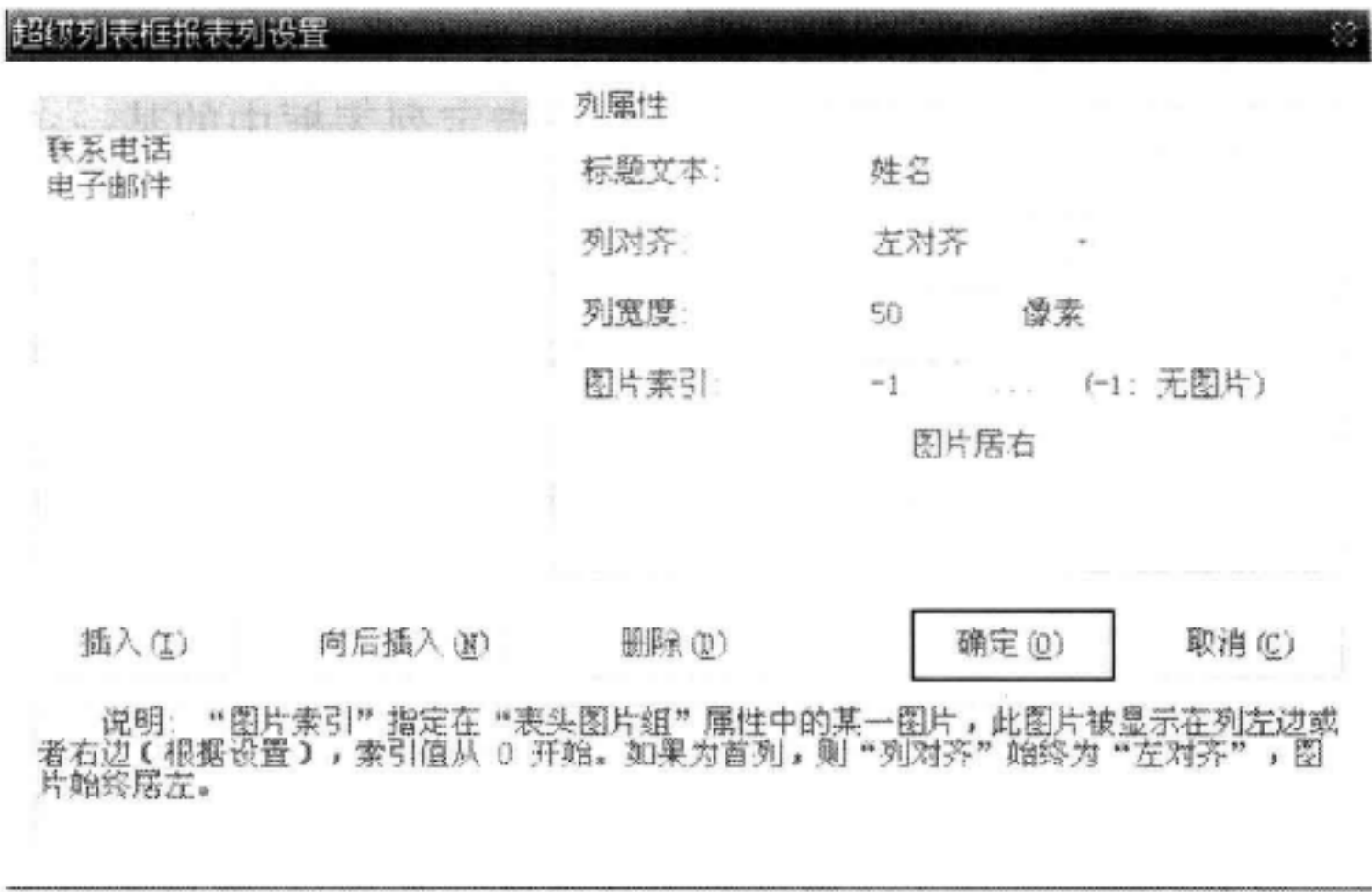


图 11-39 “报表列”设置

添加一个编辑窗口,对超级列表框表项的编辑、添加都在此窗口中进行,如图 11-43 所示。

启动窗口的程序代码如图 11-44 所示。

编辑窗口的程序代码如图 11-45 所示。

【运行结果】运行例程 11-4,观测通过右键选项对超级列表框的各表项进行编辑、插入、增加和删除的效果,图 11-46 所示为超级列表框增加了一行表项的状态。

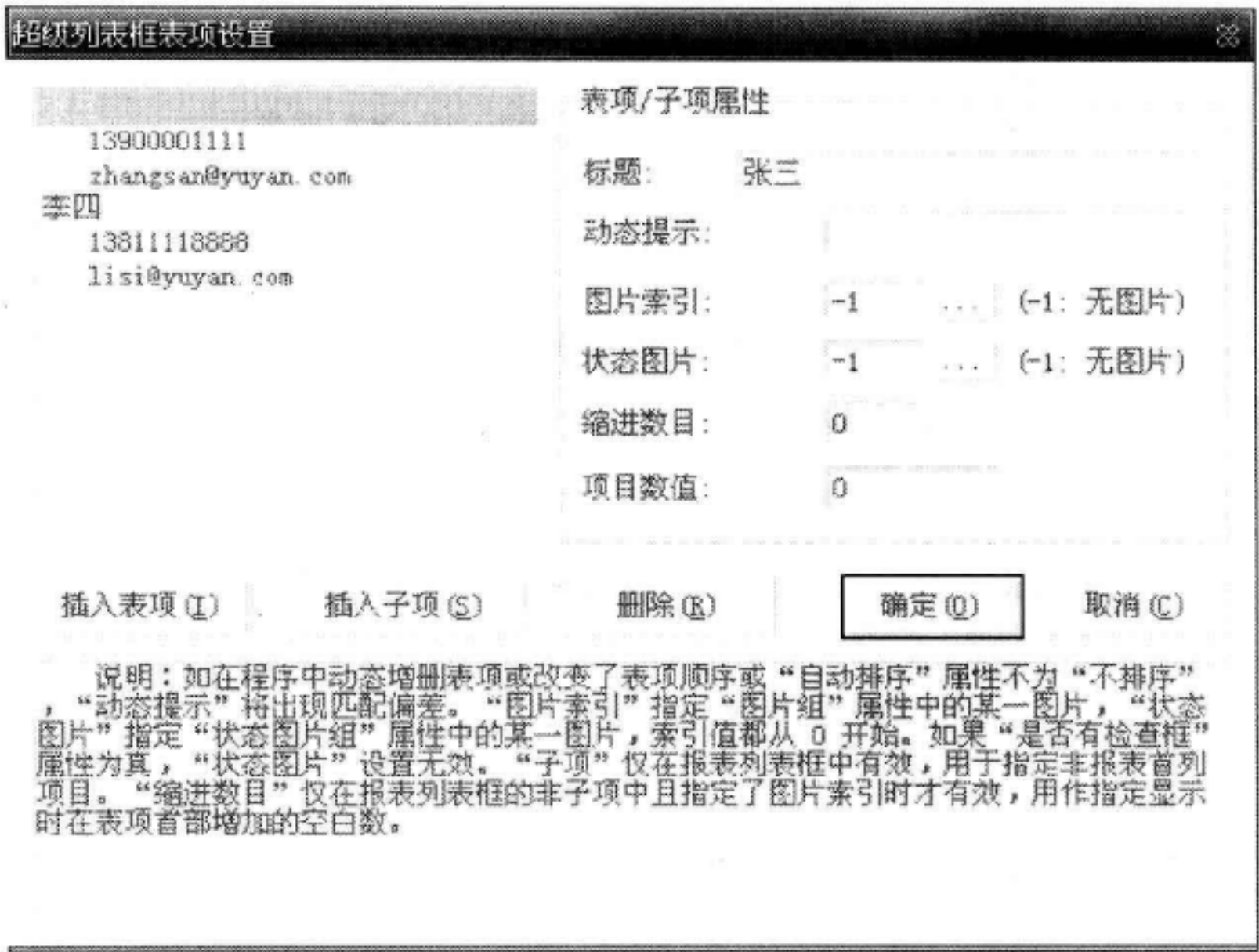


图 11-40 “表项”设置

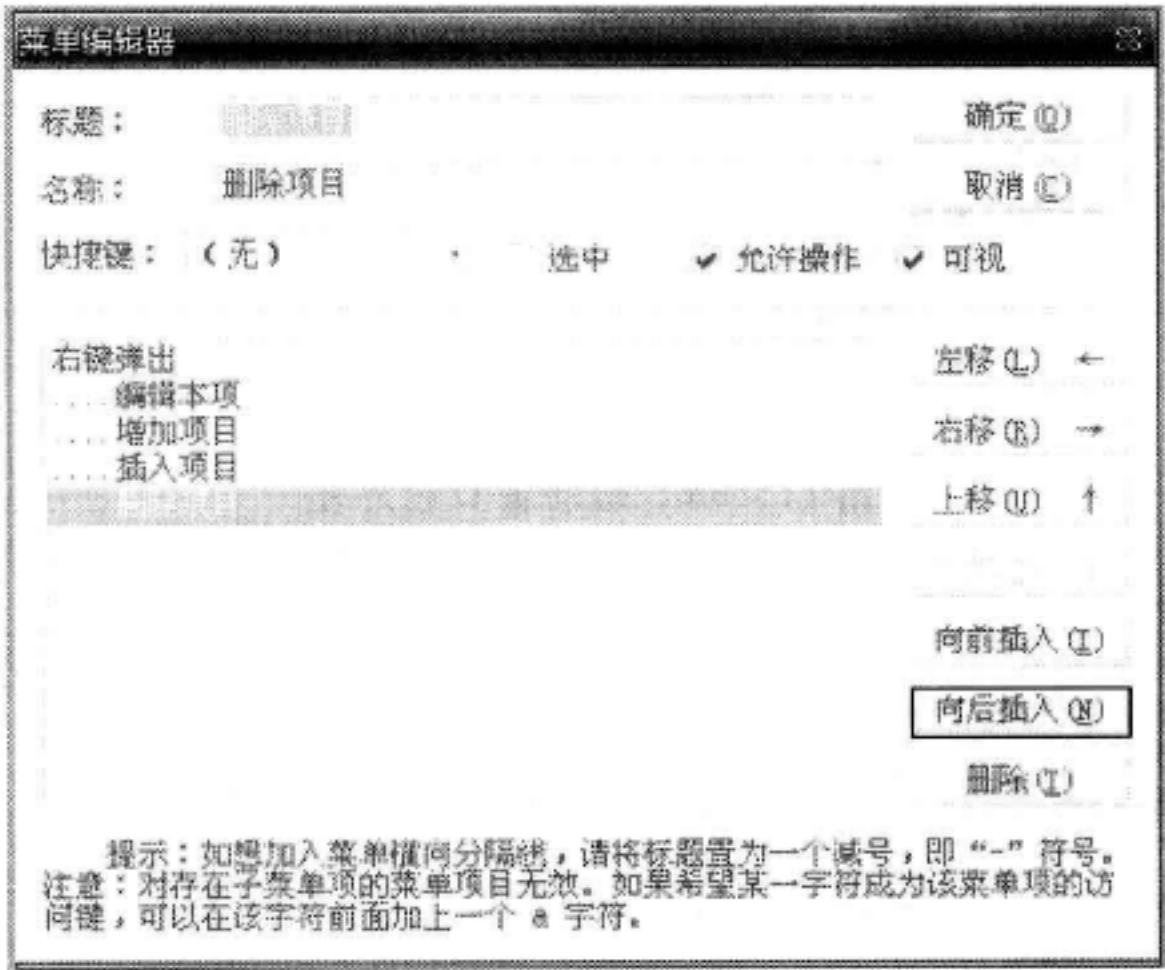


图 11-41 菜单编辑器窗口

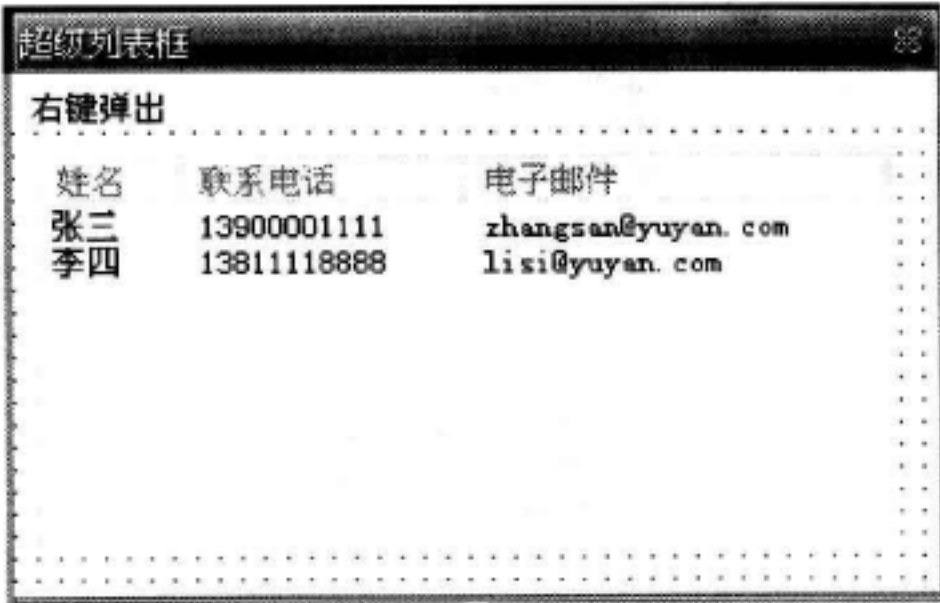


图 11-42 超级列表框设计窗口

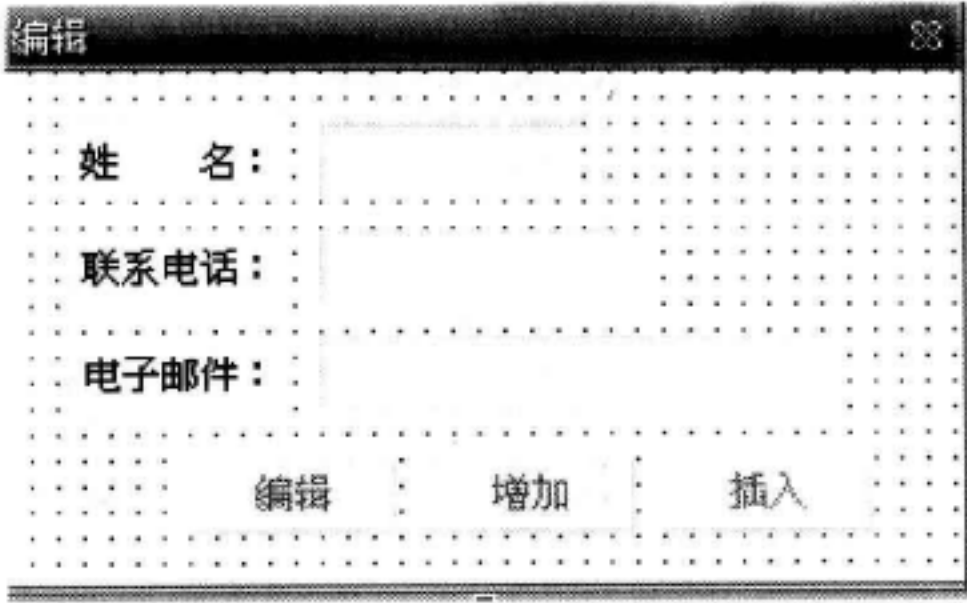


图 11-43 编辑窗口

子程序名	返回值类型	公开	备 注		
_超级列表框1_鼠标右键被按下	逻辑型				
参数名	类 型	参考	可空	数组	备 注
横向位置	整数值				
纵向位置	整数值				
功能键状态	整数值				

弹出菜单 (右键弹出, ,)

子程序名	返回值类型	公开	备 注		
_删除项目_被选择					

如果 (超级列表框1. 现行选中项 = -1)
信息框 (“请选择项目”, 0,)
超级列表框1. 删除表项 (超级列表框1. 现行选中项)

子程序名	返回值类型	公开	备 注		
_增加项目_被选择					

_启动窗口. 标记 = “增加”
如果 (超级列表框1. 现行选中项 = -1)
信息框 (“请选择项目”, 0,)
载入 (窗口1, , 假)

子程序名	返回值类型	公开	备 注		
_插入项目_被选择					

_启动窗口. 标记 = “插入”
如果 (超级列表框1. 现行选中项 = -1)
信息框 (“请选择项目”, 0,)
载入 (窗口1, , 假)

子程序名	返回值类型	公开	备 注		
_编辑本项_被选择					

_启动窗口. 标记 = “编辑”
如果 (超级列表框1. 现行选中项 = -1)
信息框 (“请选择项目”, 0,)
载入 (窗口1, , 假)

图 11-44 启动窗口程序集

子程序名	返回值类型	公开	备注
_编辑按钮_被单击			

_启动窗口.超级列表框1.置标题 (_启动窗口.超级列表框1.现行选中项, 0, 编辑框1.内容)
_启动窗口.超级列表框1.置标题 (_启动窗口.超级列表框1.现行选中项, 1, 编辑框2.内容)
_启动窗口.超级列表框1.置标题 (_启动窗口.超级列表框1.现行选中项, 2, 编辑框3.内容)
销毁 0

子程序名	返回值类型	公开	备注
_增加按钮_被单击			

变量名	类型	静态	数组	备注
新表项数	整数型			

_启动窗口.超级列表框1.插入表项 (_启动窗口.超级列表框1.取表项数 0 + 1, , , , ,)
_启动窗口.超级列表框1.现行选中项 = _启动窗口.超级列表框1.取表项数 0 + 1
_启动窗口.超级列表框1.置标题 (_启动窗口.超级列表框1.取表项数 0, 0, 编辑框1.内容)
_启动窗口.超级列表框1.置标题 (_启动窗口.超级列表框1.取表项数 0, 1, 编辑框2.内容)
_启动窗口.超级列表框1.置标题 (_启动窗口.超级列表框1.取表项数 0, 2, 编辑框3.内容)
销毁 0

子程序名	返回值类型	公开	备注
_插入按钮_被单击			

_启动窗口.超级列表框1.插入表项 (_启动窗口.超级列表框1.现行选中项 + 1, , , , ,)
_启动窗口.超级列表框1.置标题 (_启动窗口.超级列表框1.现行选中项 + 1, 0, 编辑框1.内容)
_启动窗口.超级列表框1.置标题 (_启动窗口.超级列表框1.现行选中项 + 1, 1, 编辑框2.内容)
_启动窗口.超级列表框1.置标题 (_启动窗口.超级列表框1.现行选中项 + 1, 2, 编辑框3.内容)
销毁 0

图 11-45 编辑窗口程序集



图 11-46 超级列表框例程设计窗口

【范例解析】在启动窗口选择右键菜单的某个选项,除删除本项(直接删除),其他选项被选择后都载入编辑窗口,在编辑窗口进行相应的操作,最后将操作结果回显在启动窗口的超级列表框中。

第 12 章 数据库组件

编程在很多时候需要使用大量数据,但常规情况下数据的显示、浏览是比较困难的。为了简化易语言中的数据操作,飞扬软件工作室在易 2.0 版本之后提出了“数据应用框架”的概念。

“数据应用框架”最大的特点就是把数据、数据操作、数据显示分为 3 个不同的层次。每个层次由各自的组件完成相对独立的工作,至于各层次之间的千丝万缕的联系,则由易语言在内部实现。这 3 个层次由低到高分别是:数据提供者、数据源、数据处理者。

其中,数据提供者用于存储、提供数据;数据源用于操作数据;数据处理者用于显示数据。基本上数据提供者类似于商品仓库、后勤,而数据处理者相当于前台、展示柜台,数据源相当于包装车间及运输部门。



需要注意的是,数据源仅提供操作接口,实际操作还是由数据提供者完成的。这里所说的“数据”,不仅仅是指“数据库”,它的范围更广阔,而“数据库”只是“数据”的一部分而已。

在易语言中,可充当数据提供者的组件有:通用提供者、数据库提供者、外部数据提供者等;可充当数据源的只有一个数据源组件;可充当数据处理者的组件很多,最重要的是表格组件,此外还有编辑框、标签、图片框、组合框、列表框等拥有数据源、数据列属性的组件。

数据提供者、数据源、数据处理者三者之间必须事先“关联”起来,互相协调配合,才能共同完成对数据的处理。“关联”的方法是:

- (1) 添加相应的组件,即数据处理者、数据源、数据提供者这三个组件都必须存在;
- (2) 将数据源的“数据提供者”属性设置为某个数据提供者组件;
- (3) 将数据处理者组件的“数据源”属性设置为某个数据源组件。

12.1 通用提供者组件与数据库提供者组件

通用提供者组件和数据库提供者组件属同一类组件,都可充当“数据提供者”,它们的区别如下。

(1) 通用提供者。使用内存作为数据的存储仓库,全面支持所有数据操作接口。因此必要时可以将其他类型数据提供者内的数据导入到此类型中,以全面发挥数据源对数据的操纵能力。

(2) 数据库提供者。使用数据库作为数据的存储仓库,不支持以下数据操作接口:置行高;置类型;置文本色;置背景色;置字体名;置字体尺寸;置字体属性;置边距;置文本输入格式;置对齐方式;置密码方式;合并;分解;加线条;删线条;初始尺寸同时改变列数;在中间插入行;插入列;删除列。如果想对数据库提供者中的数据进行以上操作,应该先将数据通过数据源导出到通用提供者中。

通用提供者组件的重要属性有:“初始行数”、“初始列数”属性。

“初始行数”、“初始列数”两属性均为整数型,分别用于指定初始数据的行、列数,默认值都是0。当通用提供者跟数据源、表格正确关联后,如果不设置这两个属性,表格中仍然一片空白,看不出一点表格的样子(因为初始行、列数默认值都是0)。为了美观可随便设定某个值。导入数据之前要清除这两个属性的值,清除代码为:

数据源 1. 初始尺寸(0,0)

数据库提供者组件的重要属性有:数据库文件名、字节集字段处理、数据库密码属性。

1. “数据库文件名”属性

该属性指定欲操作的数据库全路径文件名,这是数据库提供者最重要的一个属性。

2. “字节集字段处理”属性

该属性指定对字节集类型字段的处理方式。有以下可选值:0. 跳过、1. 视为图片数据、2. 视为字节集数据,默认值是0. 跳过。如果确信数据库中没有字节集字段,可置为0;如果确信有字节集字段且为图片数据,则置为1;如果确信有字节集字段但不是图片数据,可置为2。通常保持默认值0,当然置为2是最保险的。

3. “数据库密码”属性

易语言3.3以上版本提供了数据库加密功能,以保护数据库的安全。而当数据库有密码时,可以通过在“数据库密码”属性中填充密码来安全打开数据库。

【范例12-1】在设计窗体上放3个组件:数据库提供者、数据源、表格,如图12-1所示。设置数据库提供者组件的“数据库文件名”属性,即选择数据库文件(□.edb),设置数据源组件的“数据提供者”属性为“数据库提供者1”(从下拉列表中选择),设置表格组件的“数据源”属性为“数据源1”(从下拉列表中选择)。此时指定的数据库中的内容会自动显示到表格组件中。具体参考例程12-1。

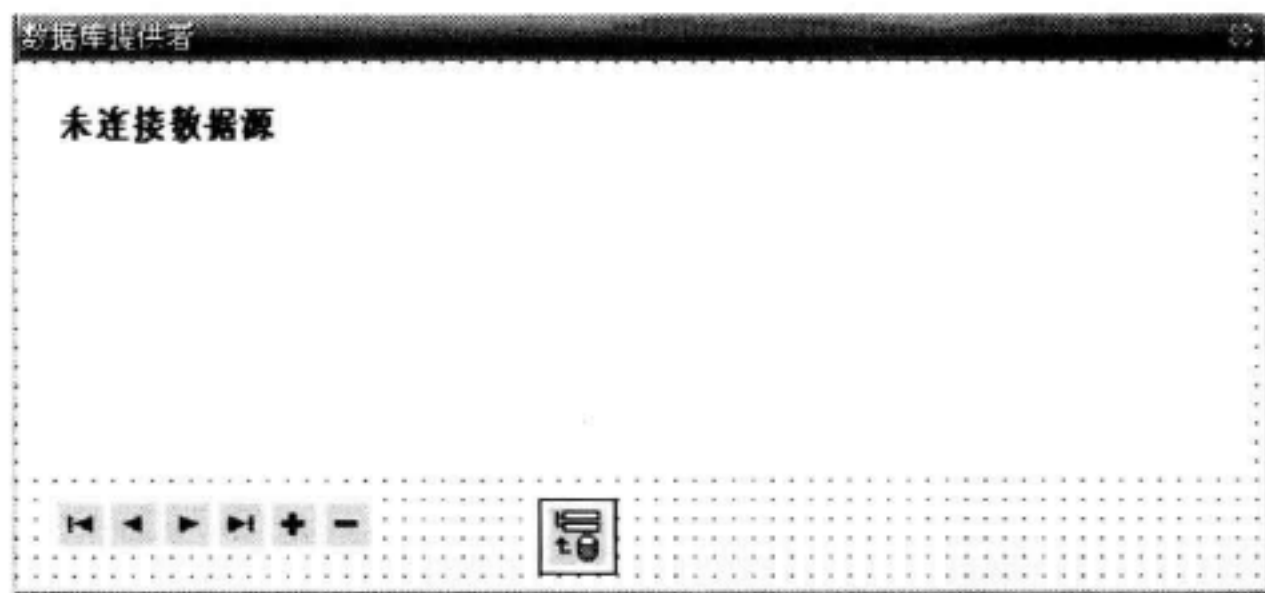


图12-1 数据库提供者组件范例窗口设计

【运行结果】运行例程12-1,数据库提供者组件在运行时不可视,运行效果如图12-2所示。如果对数据源进行操作,数据库中的内容也会被同时改动,并且表格中的数据也会同时更新。

【范例解析】对三个组件相关属性的设置是将其“关联”起来,使数据库的数据在表格中显示出来。

【范例12-2】将数据库提供者中的数据“导入”到通用数据提供者。窗口中添加的组件有:数据库提供者、通用提供者、通用对话框、数据源、表格、按钮。如图12-3所示。设置数据库提供者组件的“数据库文件名”属性,即选择数据库文件(*.edb),设置数据源组件的“数据提供者”属性为“通用提供者1”(从下拉列表中选择),设置表格组件的“数据源”属性为“数据源1”(从下拉列表中选择)。具体参考例程12-2。

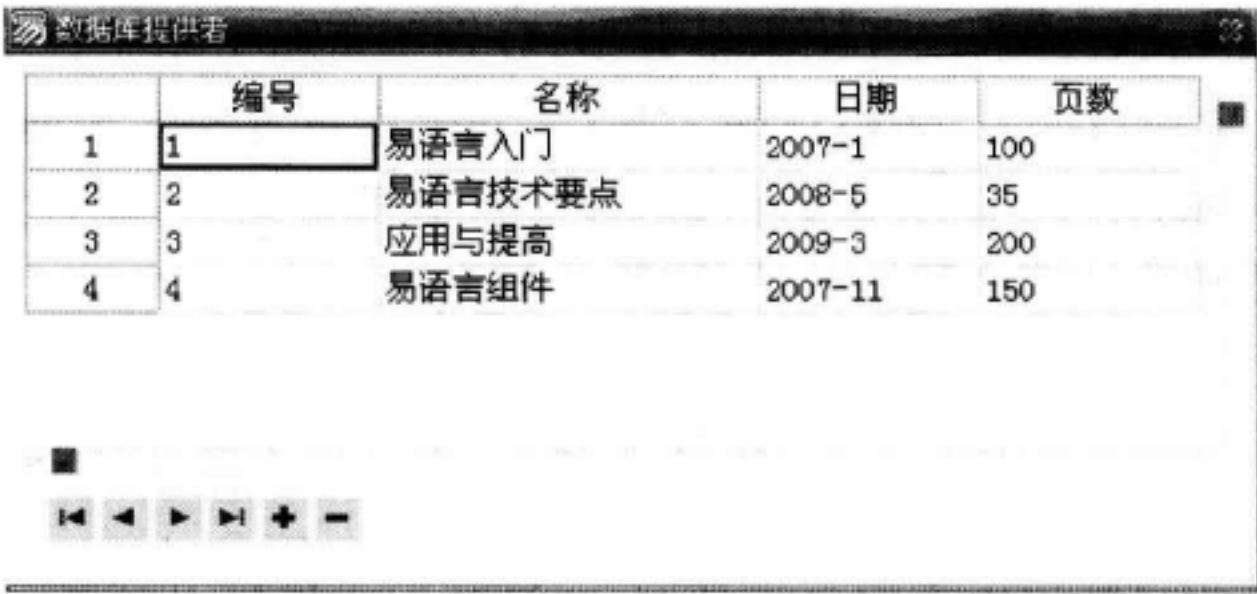


图 12-2 数据库提供者组件范例运行结果

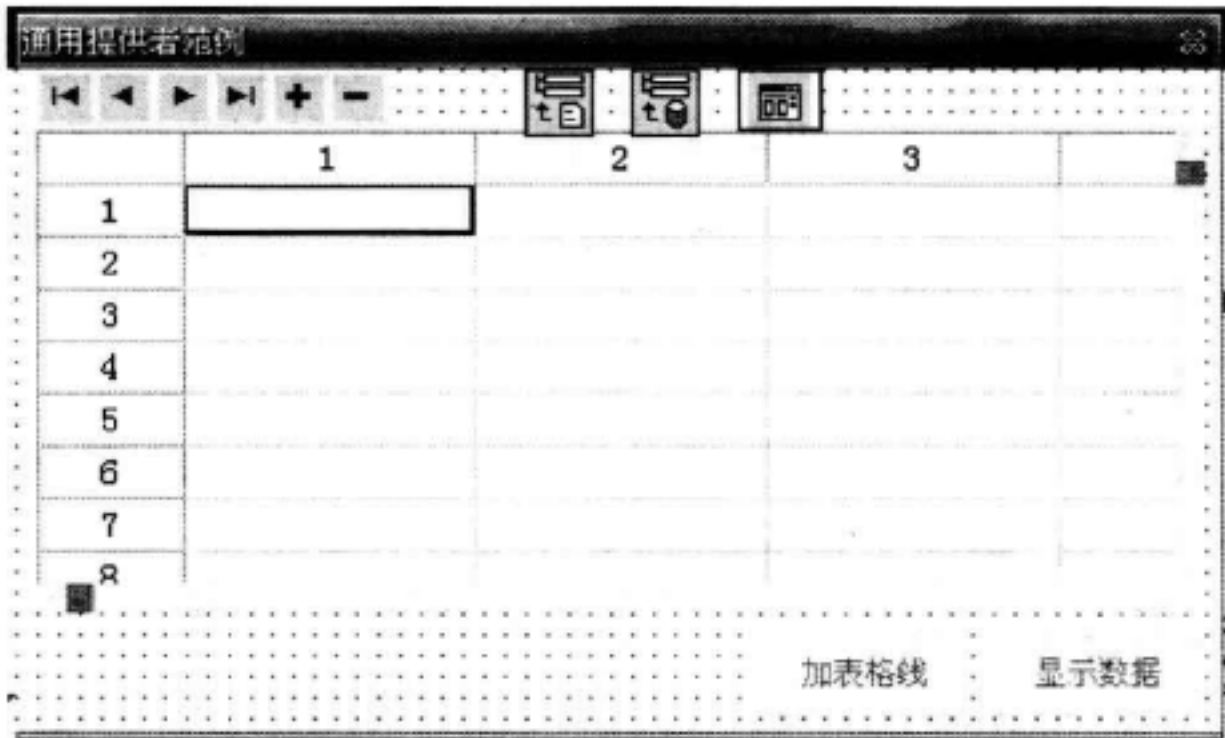


图 12-3 通用提供者组件范例窗口设计

启动窗口的程序代码如图 12-4 所示。

窗口程序集名	保留	保留	备注
窗口程序集1			

子程序名	返回值类型	公开	备注
_按钮1_被单击			

数据源1. 加线条 (0, 1, 数据源1. 取行数 0, 数据源1. 取列数 0, #左边框 + #上边框 + #右边框 + #下边框 + #水平线 + #垂直线)

子程序名	返回值类型	公开	备注
_按钮2_被单击			“导入”数据

通用对话框1. 初始目录 = 取当前目录 0
如果真 (通用对话框1. 打开 () = 真)
 数据库提供者1. 数据库文件名 = 通用对话框1. 文件名
数据源1. 初始尺寸 (0, 0)
数据源1. 添加 (数据库提供者1, ,)

图 12-4 启动窗口程序集

【运行结果】运行例程 12-2, 观测单击“加表格线”按钮和“显示数据”按钮的变化, 效果分别如图 12-5 和图 12-6 所示。

【范例解析】“加表格线”按钮的被单击事件子程序在指定单元格的四周加上边框以及单元格之间加表格线; “显示数据”按钮的被单击事件子程序将“数据库提供者 1”中的数据“导入”到数据源 1 中, 导入后数据就能显示在表格中。数据导入的方法是使用数据源组件的“添加()”方法, 代码如下:

```
数据源 1. 添加(数据库提供者 1,,)
或
数据源 1. 添加(数据源 2,,)
```

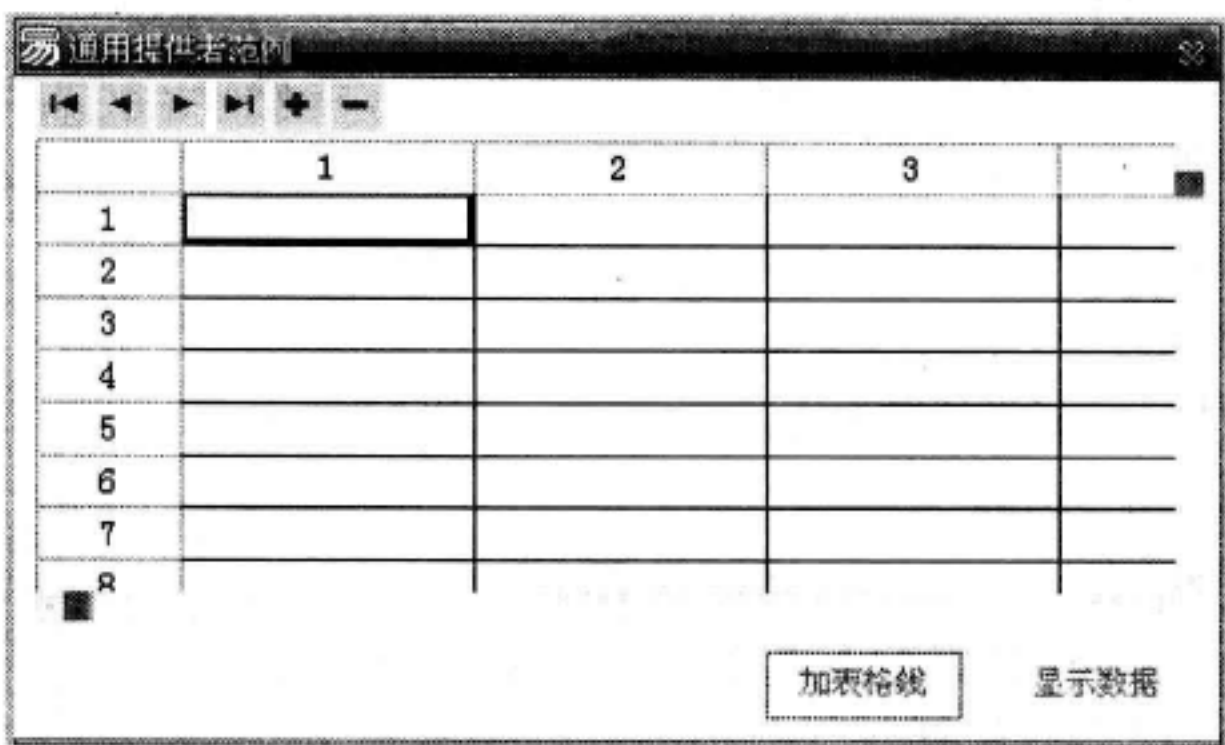



图 12-5 通用提供者组件范例运行结果 1

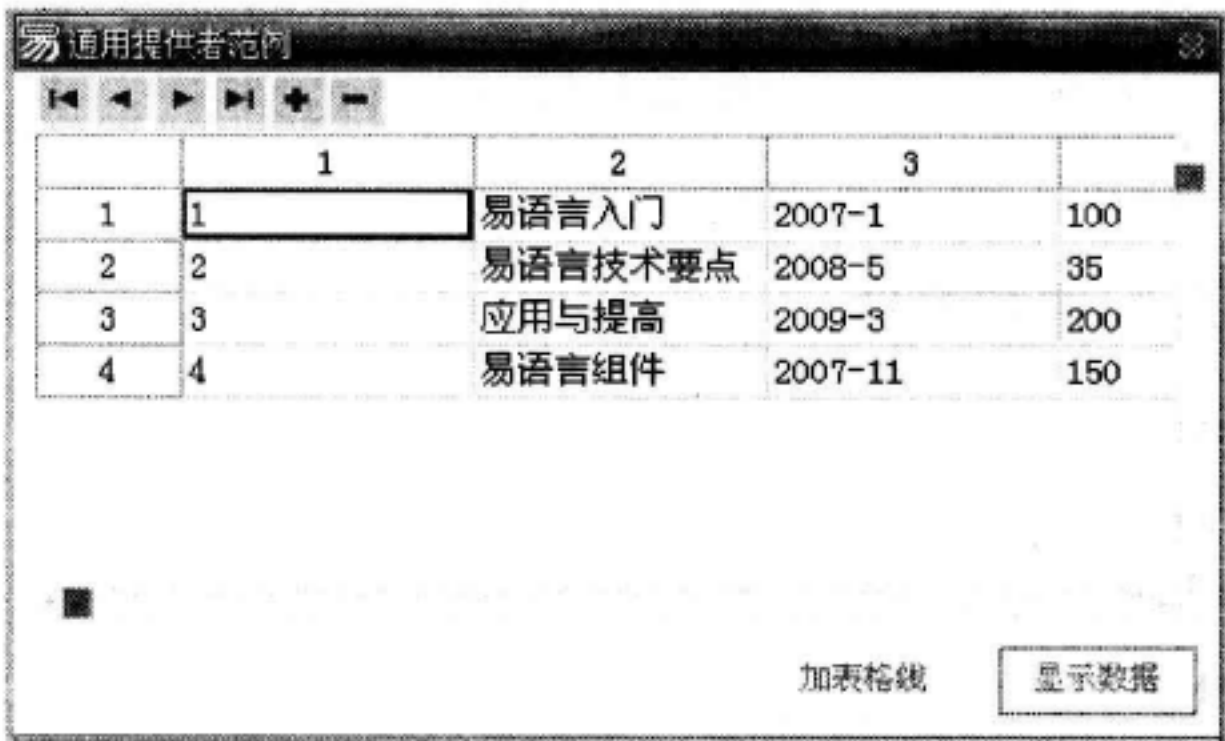



图 12-6 通用提供者组件范例运行结果 2

【提示】通用提供者组件和数据库提供者组件是最重要的两个数据提供者,前者功能强大,可对数据进行各种操作,但没有直接的数据来源;后者功能受限,只能完成数据的基本操作,但可以直接连接到数据库。二者的选用原则如下:

- (1) 如果要使用数据库,且对数据的操作不涉及外观设置(如修改字体、颜色或单元格线条等),可选用数据库提供者。
- (2) 如果需要使用数据库,又想对数据进行外观设置,可同时使用数据库提供者和通用提供者组件(先把数据库提供者中的数据“导入”通用提供者中,再对后者操作)。
- (3) 如果没有用到数据库,可单独选用通用提供者。

12.2 数据源组件

数据源组件是数据提供者与数据处理者之间的纽带,数据源组件是可视组件,如图 12-7 所示。数据源组件的 6 个按钮功能分别为到首记录、上一条记录、下一条记录、到尾记录、添加一行记录、删除一行记录。

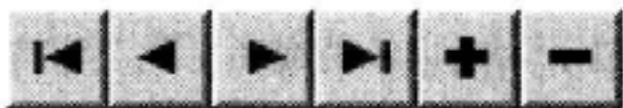


图 12-7 数据源组件

12.2.1 数据源组件的属性

数据源的重要属性有:数据提供者、只读、允许添加、允许删除属性。

1. “数据提供者”属性

该属性指定本数据源所基于的数据提供者单元名。在属性面板中设置本属性时,如果此时窗体上已经放置了某个(或多个)数据提供者组件,则会以下拉列表的形式列出,只需选择其中之一即可,如图 12-8 所示。

如果要在程序中用代码的方式为本属性赋值,只需赋数据提供者组件名称的文本形式,例如:

数据源 1. 数据提供者 = “通用数据提供者 1”

“数据提供者”属性是数据源组件最重要的属性,不设置该属性,数据源基本上没有存在的必要。

2. “只读”、“允许添加”、“允许删除”属性

这三个属性都是逻辑型的,用于限制用户对数据操作的权限。在确实需要添加或删除记录时可以调用数据源组件的如下命令:插入行、添加行、删除行、插入列、删除列等。



图 12-8 设置数据源的“数据提供者”属性

12.2.2 数据源组件的事件

数据源组件的重要事件有:当前记录被改变、添加记录、删除记录。

1. “当前记录被改变”事件

当操作者按下数据源的前 4 个按钮时产生“当前记录被改变”事件。

2. “添加记录”事件

当按下添加记录按钮添加了新记录时,即带有“+”号的按钮时产生。

3. “删除记录”事件

按下删除记录按钮删除了当前记录时,即带有“-”号的按钮时产生。

12.2.3 数据源组件的方法

数据源组件的方法非常之多,这里不可能一一介绍,查看支持库面板中的“数据类型”→“数据源”,提示面板里面将列出的数据源组件的所有属性方法和事件(显示于状态夹中)。

数据源组件的方法分类如下。

(1) 记录的操作类方法:“到首记录()”,“到尾记录()”,“跳过()”,“跳到()”,“取记录号()”。

(2) 行列的操作类方法:“取行数()”,“取列数()”,“插入行()”,“添加行()”,“删除行()”,“插入列()”,“删除列()”。

(3) 数据存取操作类方法:“置文本()”,“取文本()”,“置数据()”,“取数据()”,“添加()”,“初始尺寸()”,“存到字节集()”,“从字节集读()”,“存到文件()”,“从文件读()”,“单元格到字节集()”,“字节集到单元格()”,“单元格到文件()”,“文件到单元格()”,“刷新显示()”,“保存更改()”。

(4) 外观的操作类方法:“置表头行数()”,“置表头列数()”,“置行高()”,“置列宽()”,“置文本色()”,“置背景色()”,“置字体名称()”,“置字体尺寸()”,“置字体属性()”,“置边距()”,“置对齐方式()”,“置初始属性()”。

(5) 单元格操作类方法：“合并()”，“分解()”，“是否被合并()”，“加线条()”，“删线条()”，“是否有线条()”，“清除()”。

(6) 打印操作类方法：“打印设置()”，“置打印设置()”，“取打印设置()”，“取打印页宽()”，“取打印页高()”。

【范例 12-3】按钮模拟数据源操作。窗口中添加的组件有：数据库提供者、通用提供者、通用对话框、数据源、表格、按钮（两个命令按钮和六个数据源模拟按钮）。数据源组件的“可视”属性置为“假”。如图 12-9 所示。具体参考例程 12-3。

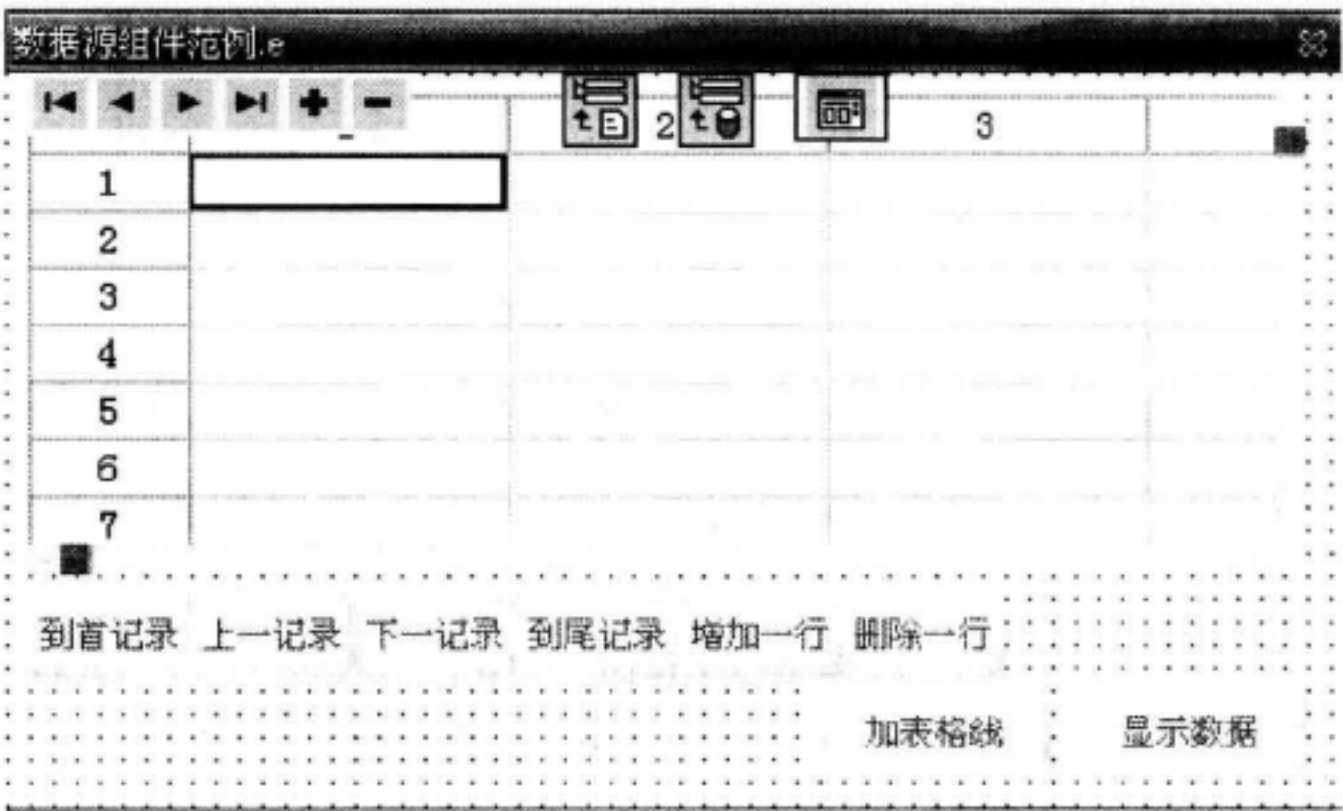


图 12-9 数据源组件范例窗口设计

与模拟按钮相关的程序代码如图 12-10 所示。

子程序名	返回值类型	公开	备注
按钮到首记录_被单击			
数据源1.到首记录 0			

子程序名	返回值类型	公开	备注
按钮到尾记录_被单击			
数据源1.到尾记录 0			

子程序名	返回值类型	公开	备注
按钮上一记录_被单击			
数据源1.跳到 (表格1.取光标行号 0 - 1)			

子程序名	返回值类型	公开	备注
按钮下一记录_被单击			
数据源1.跳到 (表格1.取光标行号 0 + 1)			

子程序名	返回值类型	公开	备注
按钮增加一行_被单击			
如果 (数据源1.数据提供者 ≠ “数据库提供者1”) 数据源1.插入行 (表格1.取光标行号 0,) 数据源1.插入行 (数据源1.取行数 0 + 1,) 表格1.置光标 (数据源1.取行数 0, 1)			

子程序名	返回值类型	公开	备注
按钮删除一行_被单击			
数据源1.删除行 (表格1.取光标行号 0, 表格1.取选择行数 0)			

图 12-10 启动窗口程序集

【运行结果】运行例程 12-3,观测单击各个模拟按钮后表格中数据的变化,如图 12-11 所示。




图 12-11 数据源组件范例运行结果

【范例解析】“到首记录”按钮与“到尾记录”按钮的程序代码是使用了数据源的两个方法命令。“上一记录”按钮与“下一记录”按钮的程序代码没有使用数据源组件的取行号命令,而是使用了表格组件的取行号命令,这是因为表格如果添加了空白行的话,它的行号比数据源中的记录号要多,因此取表格的光标行号要更加准确一些。“增加一行”按钮使用了取数据源中的所有记录行数命令,即为数据源的末尾加空白行。而“删除一行”按钮使用了取表格光标行号的命令,这样在光标处就会被删除。

【提示】因为数据源组件上的添加删除记录的按钮不利于数据的完全操作。在实际应用中,通常把数据源组件的“可视”属性置为“假”,使它对用户不可见,令用户无法直接操作数据。

12.3 表格组件

表格组件  是以行和列的形式显示数据,一般与数据提供者和数据源组件联用。

12.3.1 表格组件的属性

表格组件的属性如图 12-12 所示。

1. “数据源”属性

指定与表格相关联的数据源组件。在属性夹中设置本属性时,如果窗体上已经放置了某个(或多个)数据源组件,则会以下拉列表的形式列出,选择其中之一即可。如果要在程序中用代码的方式为本属性赋值,只需赋数据源组件名称的文本形式即可,例如:

表格 1. 数据源 = “数据源 1”

“数据源”属性是表格组件最重要的属性,如果不设置该属性,表格基本上没有存在的必要。

2. “缩放比”属性

指定表格在显示数据时所采用的显示比例,可以是 20 ~ 1000 内的任意整数值,默认值是 100。

如果要以页面的等宽显示,可以直接使用如下方法:

表格 1. 等宽缩放()

3. “允许选择块”属性

本属性为逻辑型。指定是否允许操作者选择表格单元格区域。

如果为“真”，即可以在表格中拖动鼠标左键，同时选中多个单元格，如图 12 - 13 所示。
如果为“假”，即只能选择一个单元格。



图 12 - 12 表格组件属性夹

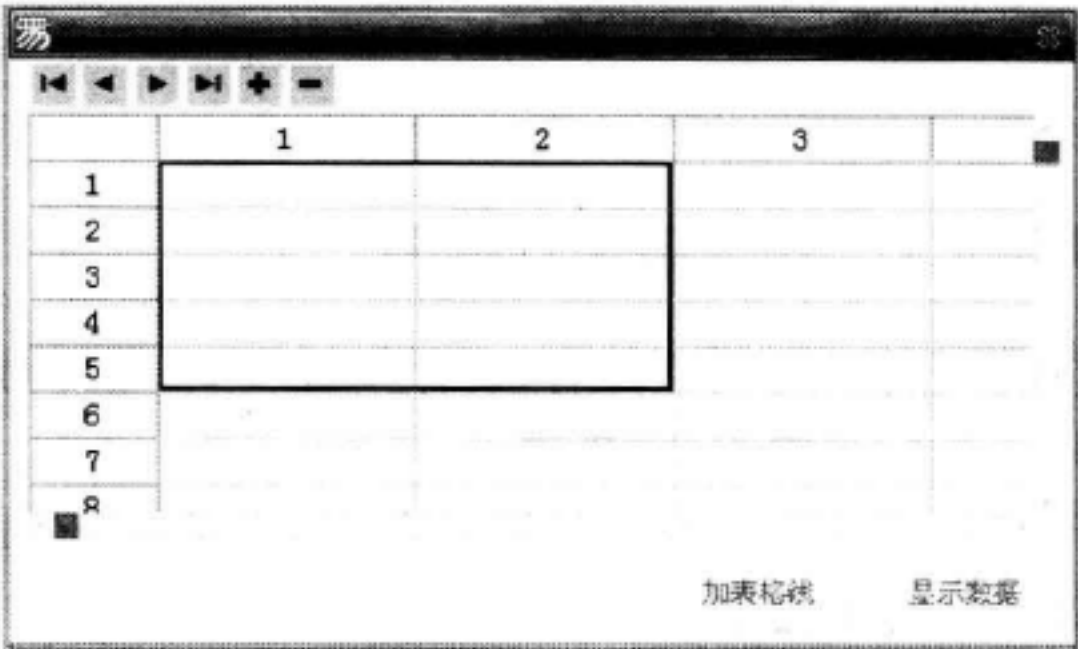


图 12 - 13 “允许选择块”属性为“真”

4. “显示标尺”属性

本属性为逻辑型。用于控制是否显示表格最上面一行和最左面一列的标尺。本属性设置为“假”时，如图 12 - 14 所示。

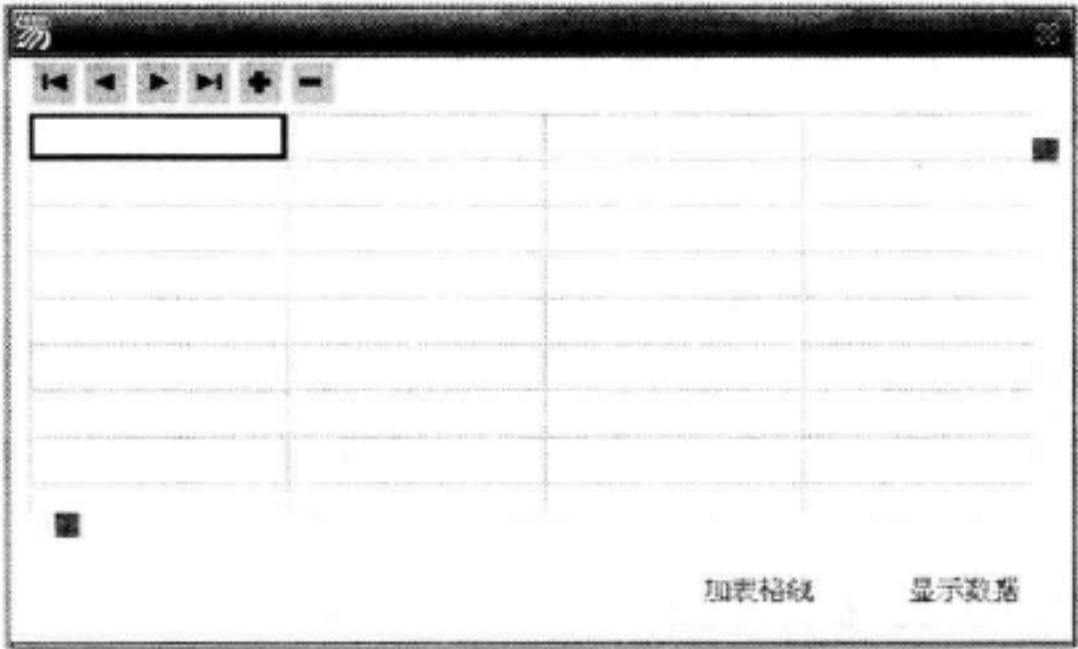


图 12 - 14 “显示标尺”属性为“假”

5. “显示空表格线”属性

本属性为逻辑型。指定是否用虚线显示空表格线。本属性设置为“假”时，表格显示的效果如图 12 - 15 所示。

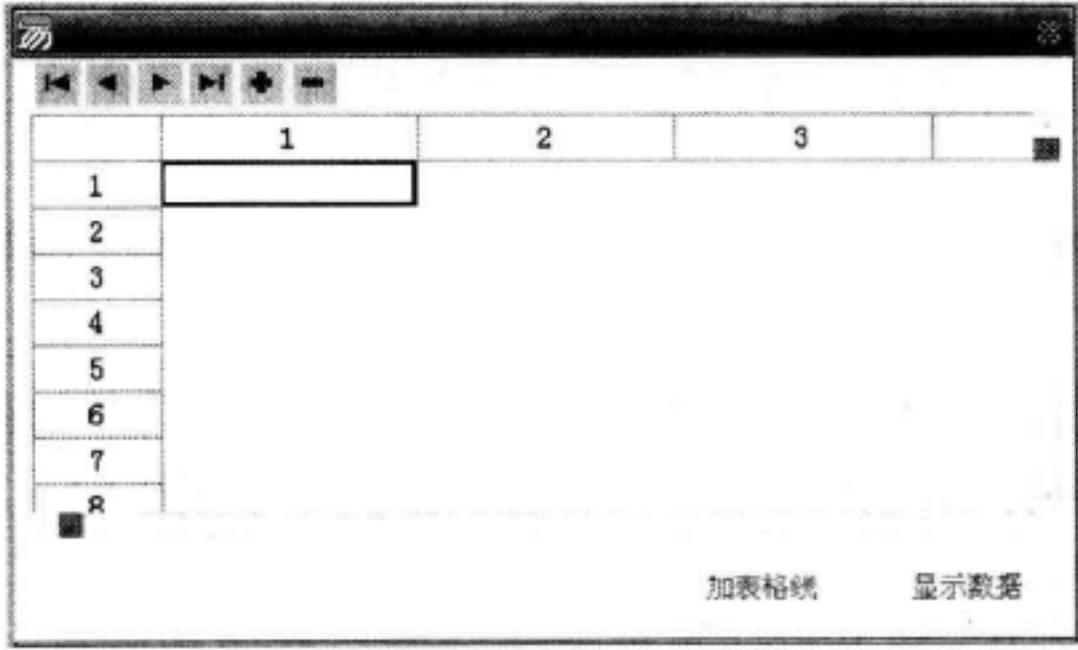


图 12 - 15 “显示空表格线”属性为“假”

6. “禁止调整行高”、“禁止调整列宽”属性

本属性为逻辑型。指定是否禁止操作者调整表格行高、列宽。

如果本属性为“真”，即为禁止，当鼠标移动到标尺时，即不会出现双向拉伸箭头。

7. “允许粘贴扩展”属性

指定当粘贴表格数据时，如果现行表格尺寸无法容纳，是否允许自动扩展。例如当本属性为“真”时，为一个 2×2 单元格的表格插入一个 4×4 单元格的表格，会自动增加不足的单元格。

12.3.2 表格组件的事件

表格组件的常用事件有：光标位置改变、选择行列数改变、内容被改变。

1. “光标位置改变”事件

当操作者改变了光标位置后会产生“光标位置改变”事件。

2. “内容被改变”事件

当操作者修改了单元格内容时会产生“内容被改变”事件。

3. “选择行列数改变”事件

当操作者更改了当前被选择区域的行列数时会产生“选择行列数改变”事件。

12.3.3 表格组件的方法

表格的专有方法有 14 个。可以在支持库面板中的“系统核心支持库”→“数据类型”→“表格”中找到，提示面板里面将列出表格组件所有的属性方法和事件（显示于状态夹中），如图 12-16 所示。

1. “取光标行号()”、“取光标列号()”方法

此类方法的功能是取表格中“光标所在的单元格”所处的行号或列号。如果调用这两个方法时，已经有多个单元格被同时选择，则返回所有被选择的单元格中最左上角的那个单元格的行号或列号。

2. “取选择行数()”、“取选择列数()”方法

此类方法的功能是取表格中被选择的单元格的行数或列数。

3. “等宽缩放()”方法

此方法的功能是缩放表格内容，使其正好完全显示在表格中。

4. “置光标()”方法

此方法的功能是定位光标到某行某列。此方法的参数是整数型的行号和列号。

5. “选择()”、“全部选择()”、“复制()”、“全部复制()”、“粘贴()”、“粘贴到光标处()”方法

这类方法是对单元格的选择、复制、粘贴操作，需要指定欲操作的一个或多个相邻的单元格。

例如可以在程序中使用一个弹出菜单操作这些命令，菜单设置如图 12-17 所示。

通过在表格上单击鼠标右键弹出菜单，具体的程序代码如图 12-18 所示。

6. “打印()”、“打印预览()”方法

此类方法的功能是打印表格及打印表格前的预览。打印预览是一个十分好的功能，可以在打印前得到实际打印效果。

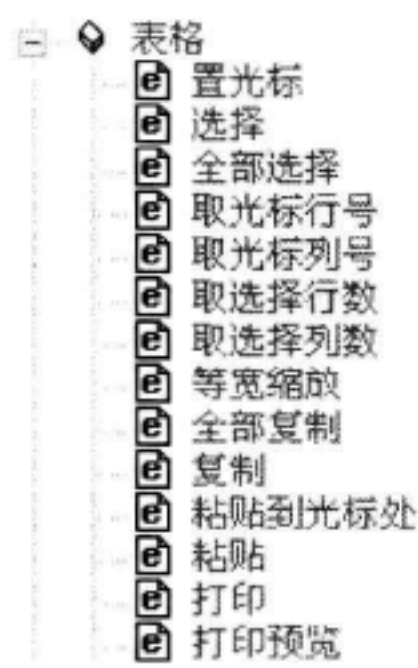


图 12-16 表格组件的方法

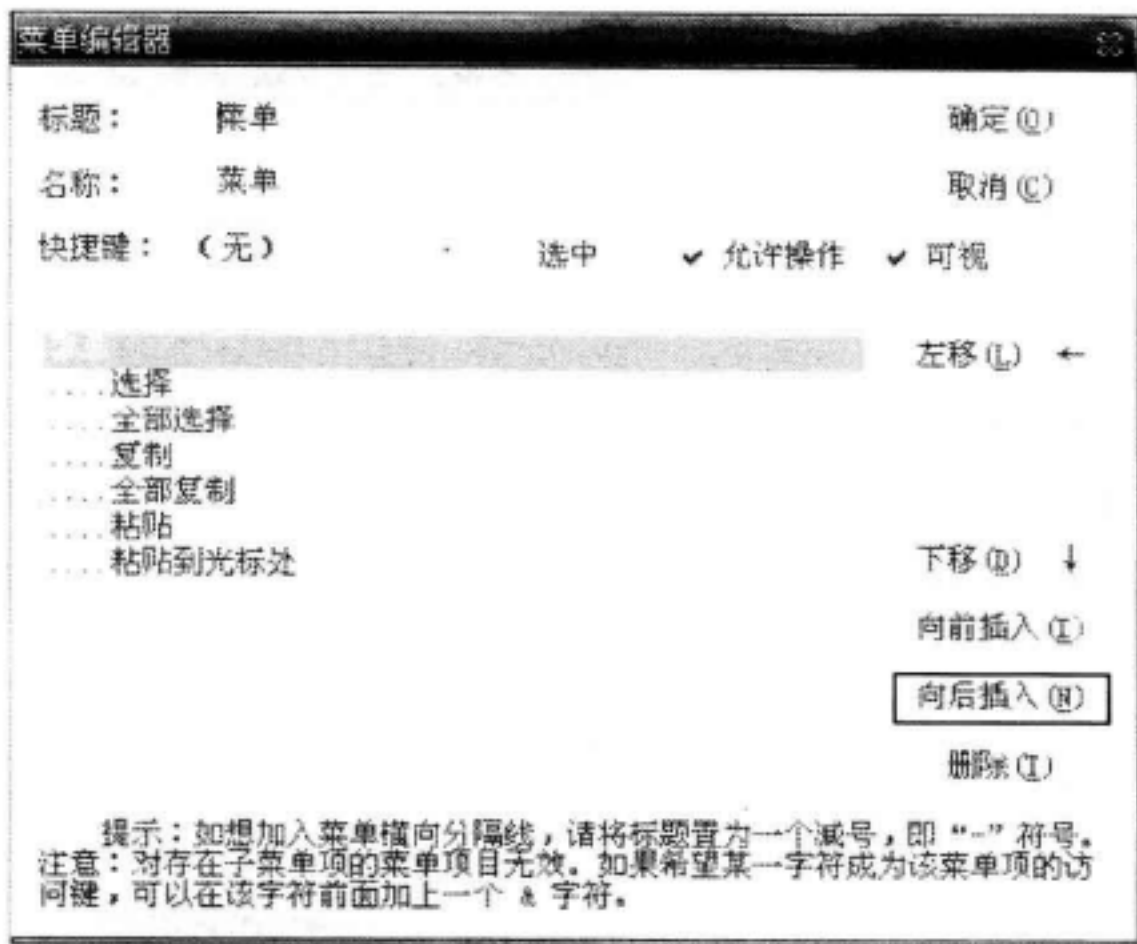


图 12-17 设置右键菜单

子程序名	返回值类型	公开	备注		
_表格1_鼠标右键被按下	逻辑型				
参数名	类型	参考	可空	数组	备注
横向位置	整型				
纵向位置	整型				
功能键状态	整型				

弹出菜单 (菜单, ,)

子程序名	返回值类型	公开	备注
_选择_被选择			

表格1. 选择 (表格1. 取光标行号 0, 表格1. 取光标列号 0, 表格1. 取选择行数 0, 表格1. 取选择列数 0)

子程序名	返回值类型	公开	备注
_全部选择_被选择			

表格1. 全部选择 0

子程序名	返回值类型	公开	备注
_复制_被选择			

表格1. 复制 (表格1. 取光标行号 0, 表格1. 取光标列号 0, 表格1. 取选择行数 0, 表格1. 取选择列数 0)

子程序名	返回值类型	公开	备注
_全部复制_被选择			

表格1. 全部复制 0

子程序名	返回值类型	公开	备注
_粘贴_被选择			

表格1. 粘贴 (表格1. 取光标行号 0, 表格1. 取光标列号 0)

子程序名	返回值类型	公开	备注
_粘贴到光标处_被选择			

表格1. 粘贴到光标处 0

图 12-18 右键菜单程序代码

【范例 12-4】表格组件模板的应用。在启动窗口添加一个表格组件、一个数据源组件和一个通用提供者组件,三者数据相连。设计窗口如图 12-19 所示。具体参考例程 12-4。启动窗口的程序代码如图 12-20 所示。



图 12-19 表格范例设计窗口

子程序名	返回值类型	公开	备注
填充基本情况1			

置当前库 (“学生”)
数据源.数据提供者 = “基本情况提供者”
数据源.置文本 (3, 2, 学号)
数据源.置文本 (3, 4, 姓名)
数据源.置文本 (3, 6, 选择 (读 (“性别”), “男”, “女”))
数据源.置文本 (4, 2, 到文本 (读 (“年龄”)))
数据源.置文本 (4, 4, 到文本 (读 (“身高”)))
数据源.置文本 (4, 6, 到文本 (读 (“体重”)))
数据源.置文本 (5, 2, 读 (“家庭住址”))
数据源.置数据 (7, 1, 读 (“像片”))
数据源.置文本 (7, 4, 读 (“期中综合评价”))
数据源.置文本 (16, 1, 读 (“期末综合评价”))
数据源.置文本 (16, 4, 读 (“年度综合评价”))

子程序名	返回值类型	公开	备注
__启动窗口_创建完毕			

变量名	类型	静态	数组	备注
学科数	整数型			
索引	整数型			

打开 (“学生”, “学生”, , , ,)
数据源.数据提供者 = “基本情况提供者”
如果真 (数据源.从文件读 (“base.grd”) = 假) : 这次从文件读入表格样板到基本情况提供者, 用作演示模板的不同来源
 信息框 (“找不到学生基本情况表格模板!”, 0, “错误”)
 销毁 0
 返回 0
学生表格.表格线颜色 = #红色
填充基本情况1 0

子程序名	返回值类型	公开	备注
__颜色选择器1_颜色被改变			

学生表格.表格线颜色 = 颜色选择器1.颜色

图 12-20 启动窗口程序集

【运行结果】运行例程 12-4, 观测表格的显示状态, 如图 12-21 所示。

【范例解析】运行后, 在表格中显示学生数据库的记录内容, 表格组件套用了—个表格模板。窗口下方的颜色选择器可以控制表格的线条颜色, 如图 12-22 所示。

易

学生基本情况表

学号:		姓名:		性别:	男
年龄:	15	身高:	1.4	体重:	60
家庭住址:	北京市				
相片			期中综合评价		
			不错 有培养价值		
期末综合评价			年度综合评价		
很好 有发展潜力			良好 需要继续努力		


选择表格颜色

关闭

图 12-21 表格组件范例运行结果 1

易

学生基本情况表

学号:		姓名:		性别:	男
年龄:	15	身高:	1.4	体重:	60
家庭住址:	北京市				
相片			期中综合评价		
			不错 有培养价值		
期末综合评价			年度综合评价		
很好 有发展潜力			良好 需要继续努力		

选择表格颜色


关闭

图 12-22 表格组件范例运行结果 2

12.4 外部数据库

除了易语言本身的数据库(文件后缀名为 edb),其他数据库系统称为外部数据库,比如微软的 Access 数据库(文件后缀名为 mdb)、dBase 数据库(文件后缀名为 dbf)、Visual FoxPro 数据库(文件后缀名为 dbc)、MS SQL Server 大型数据库、MySQL 大型数据库等。由于微软的 Access数据库是目前应用最广泛的个人桌面型数据库,所以在外部数据库应用讲座中就以 Access数据库为例。

与易数据库不同的是,Access 数据库内可以有多个表。比如一个学生管理系统数据库内包含学生成绩表、学生基本情况表、管理人员表等多个表。这样就很方便地将相关的字段集中在一起管理,各个表之间又可以相互关联。

从易语言 2.0 版开始,易语言增加了“外部数据库”组件,用作支持使用 ODBC 直接对其他所有类型数据库进行操作;2.1 版又增加了“外部数据提供者”组件,可以直接将外部数据库绑定到数据源。有了这两个组件,对于其他类型的数据库(非易语言数据库 *.edb,如 Access,Paradox,SQL,Oracle 等)也可进行直接操作。

12.4.1 外部数据库组件的属性

外部数据库组件只有简单的“名称”、“备注”、“左边”、“顶边”、“宽度”、“高度”和“标记”属性,没有专有事件,它也是一个不可视组件。

12.4.2 外部数据库组件的方法

外部数据库的重要方法有很多,在支持库面板中能查找外部数据库组件的方法。如图 12-23 所示。

1. “打开()”、“关闭()”方法

此类方法用于打开或关闭指定的 ODBC 数据源。如:

外部数据库 1. 打开(,)

外部数据库 1. 关闭()

2. “查询()”、“执行()”方法

此类方法分别用于执行查询类和非查询类 SQL 语句。例如以下两个例子:

(1) 记录集句柄 = 外部数据库 1. 查询(“select * from books”)

此语句是对当前被打开的数据库进行数据查询(检索 books 表中的所有记录的所有字段),将查询结果的记录集的句柄,赋值给整数型变量“记录集句柄”。

【注意】不用本记录集时,应将本记录集关闭。通过调用本组件的“关闭记录集()”方法实现,例如:

外部数据库 1. 关闭记录集(记录集句柄)

(2) 外部数据库 1. 执行(“insert into books values(1,2,3)”)

此语句是将记录插入到打开的数据库表 books 中。

3. “读()”方法

此方法的功能是读取指定记录集的当前记录处指定字段的数据内容。返回数据的数据类型与原外部数据库中的数据类型相对应。如:

x = 外部数据库 1. 读(记录集句柄,1)

此语句是用于读取指定记录集中当前记录的第“1”个字段。

x = 外部数据库 1. 读(记录集句柄,“序号”)

此语句用于读取指定记录集中当前记录的名称为“序号”的字段。



图 12-23 外部数据库组件的方法

“记录集句柄”是整数型变量,是在调用外部数据库组件的“查询”方法时返回的值。通常先将当前记录指针移动到某记录上,然后再“读()”。

4. “到首记录()”、“到尾记录()”、“到前一记录()”、“到后一记录()”方法

以上方法用于移动记录集的当前记录指针。这 4 个方法的语法是相同的,都有一个整数型参数,指定要操作的记录集;都有一个逻辑型参数,指示本方法是否执行成功。例如:

```
到首记录(记录集句柄)
到尾记录(记录集句柄)
到前一记录(记录集句柄)
到后一记录(记录集句柄)
```

“记录集句柄”是整数型变量,是调用外部数据库组件的“查询”方法时的返回值。

5. “首记录前()”、“尾记录后()”方法

此类方法的功能是判断当前记录指针是否在第一个记录的前面或最后一个记录的后面。例如:

```
如果(外部数据库 1. 首记录前(记录集句柄))
.....
如果结束
如果(外部数据库 1. 尾记录后(记录集句柄))
.....
如果结束
```

本例中的“记录集句柄”是整数型变量,是调用外部数据库组件的“查询”方法时的返回值。

【范例 12 - 5】使用外部数据库组件读取 Access 数据库中的记录。启动窗口中添加一个外部数据库组件,添加数个编辑框组件以显示数据库中的记录,添加几个操作按钮,设计窗口如图 12 - 24 所示。具体参考例程 12 - 5。

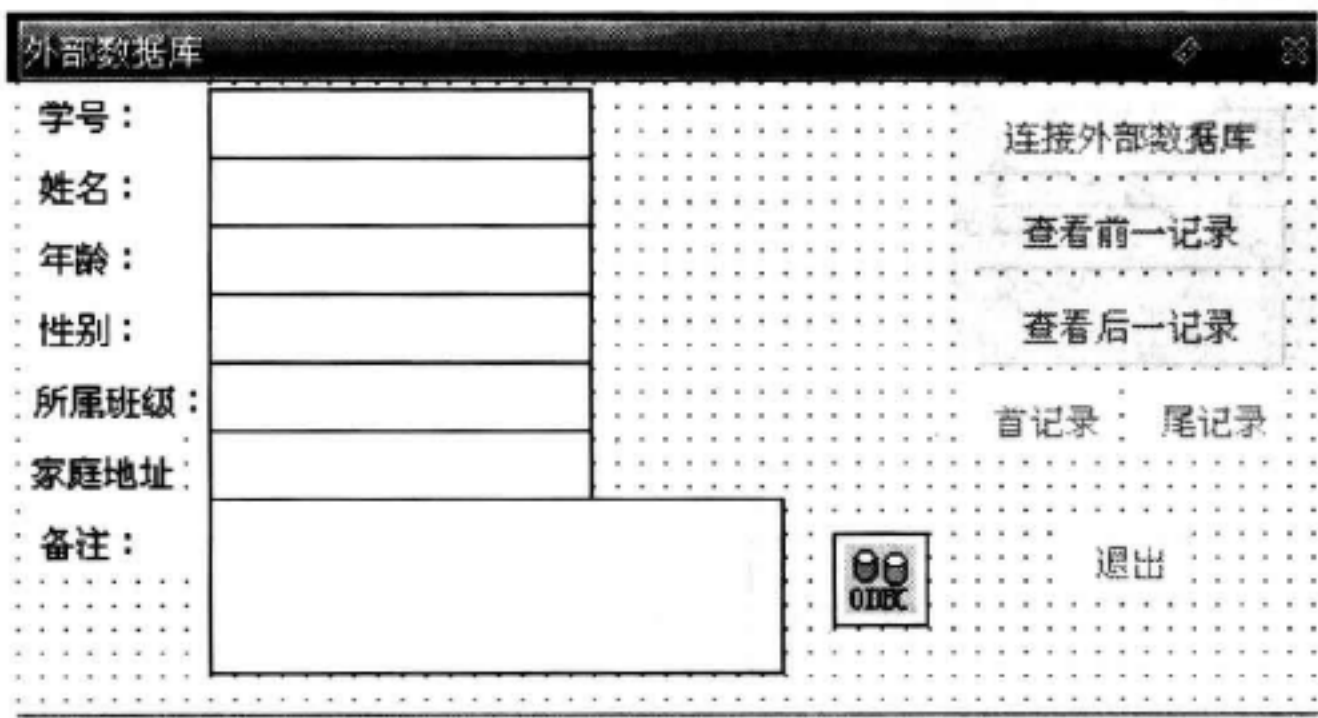


图 12 - 24 外部数据库设计窗口

连接外部数据库首先要配置 ODBC 数据源,本例中执行外部数据库 1. 打开(,)时即可手动配置,具体步骤如下:

- (1) 在弹出的选择数据源对话框(如图 12 - 25 所示)中选择其中的“新建”按钮,出现一个对话框,如图 12 - 26 所示。
- (2) 选中列表框第 2 项(Access 的驱动程序)。然后单击“下一步”按钮,在弹出的对话框

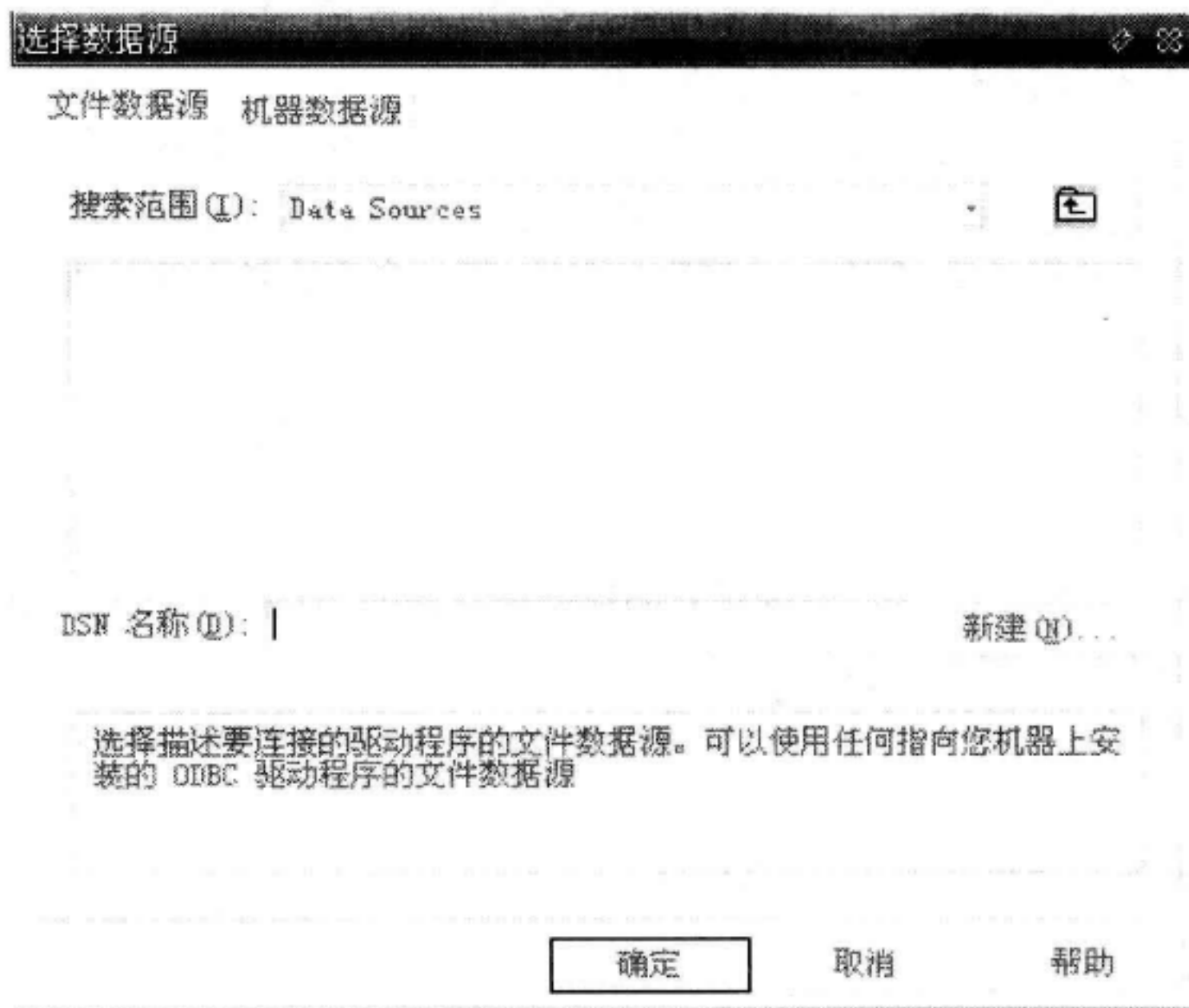


图 12-25 选择数据源窗口



图 12-26 创建新数据源窗口

中要求输入一个带路径的文件名。单击“浏览”按钮，在弹出的“另存为文件对话框”中选择一个路径，填写一个文件名，单击“保存”按钮关闭“另存为文件对话框”，会弹出如图 12-27 所示的对话框。

(3) 单击“下一步”按钮，出现图 12-28 所示对话框，单击“完成”按钮就新建了一个 DSN 文件。

(4) 此时新建的 *.dsn 文件已设置了路径和驱动程序信息，但还没有和具体的数据库相连接。在图 12-28 中单击“完成”按钮，会弹出如图 12-29 所示的对话框，在这个对话框中可



图 12-27 新数据源地址窗口



图 12-28 显示设置信息对话框

以连接数据库。

(5) 单击图中的“选择”按钮,在“打开文件”对话框中选择 Access 数据库文件“*.mdb”,最后单击“确定”按钮。这样,新建的数据源连接文件*.dsn 就包含了访问数据库所需的所有信息,利用这个文件可以在易语言中任意操纵数据库了。如图 12-30 所示。

(6) 选中图 12-30 中新建的文件*.dsn,单击“确定”按钮,会出现一个对话框,如图 12-31 所示。

图中的“数据库:...\...\学生.mdb”,正是将要连接的数据库,单击“确定”按钮完成设置。



图 12-29 ODBC Microsoft Access 安装对话框

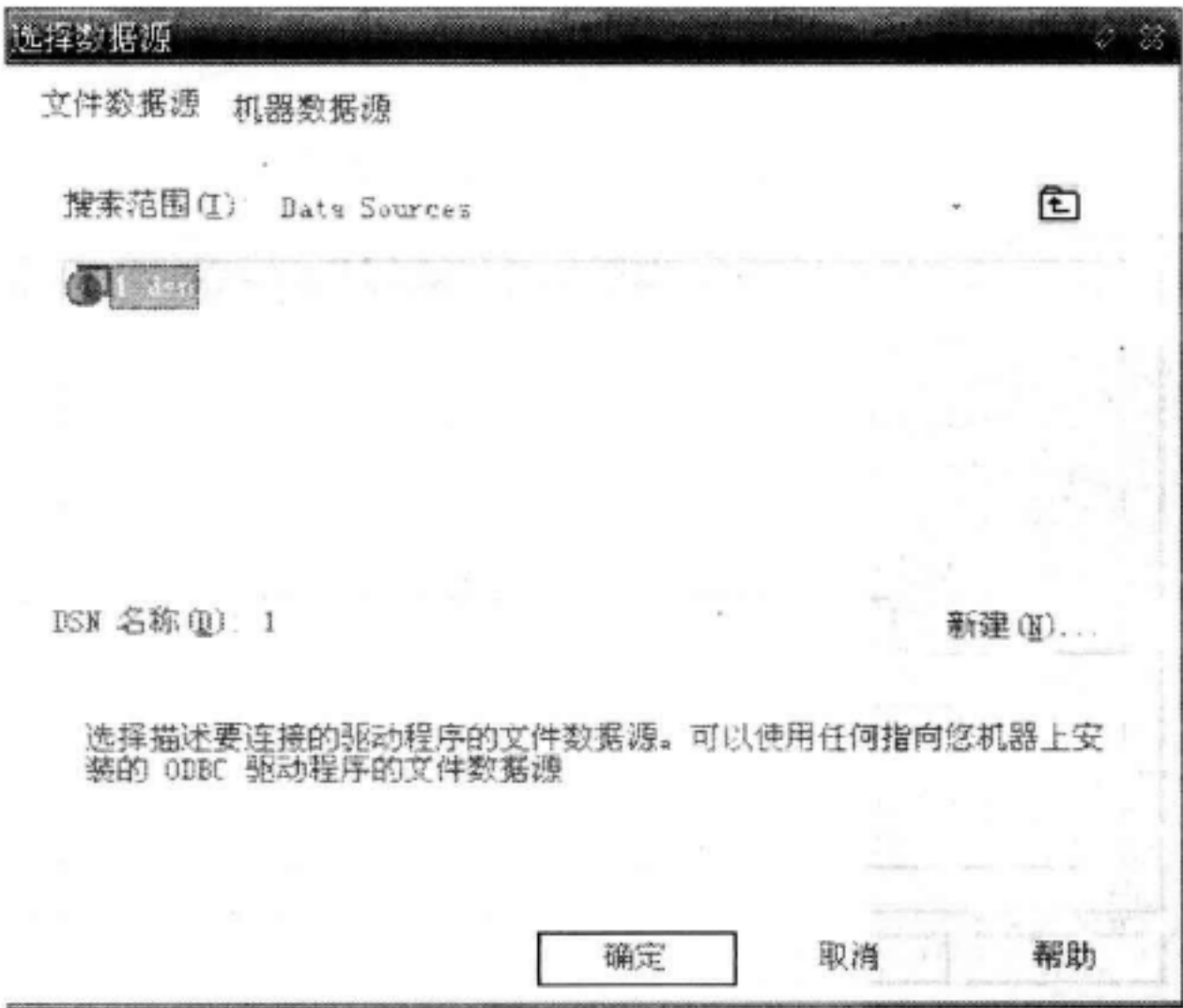


图 12-30 选择数据源窗口

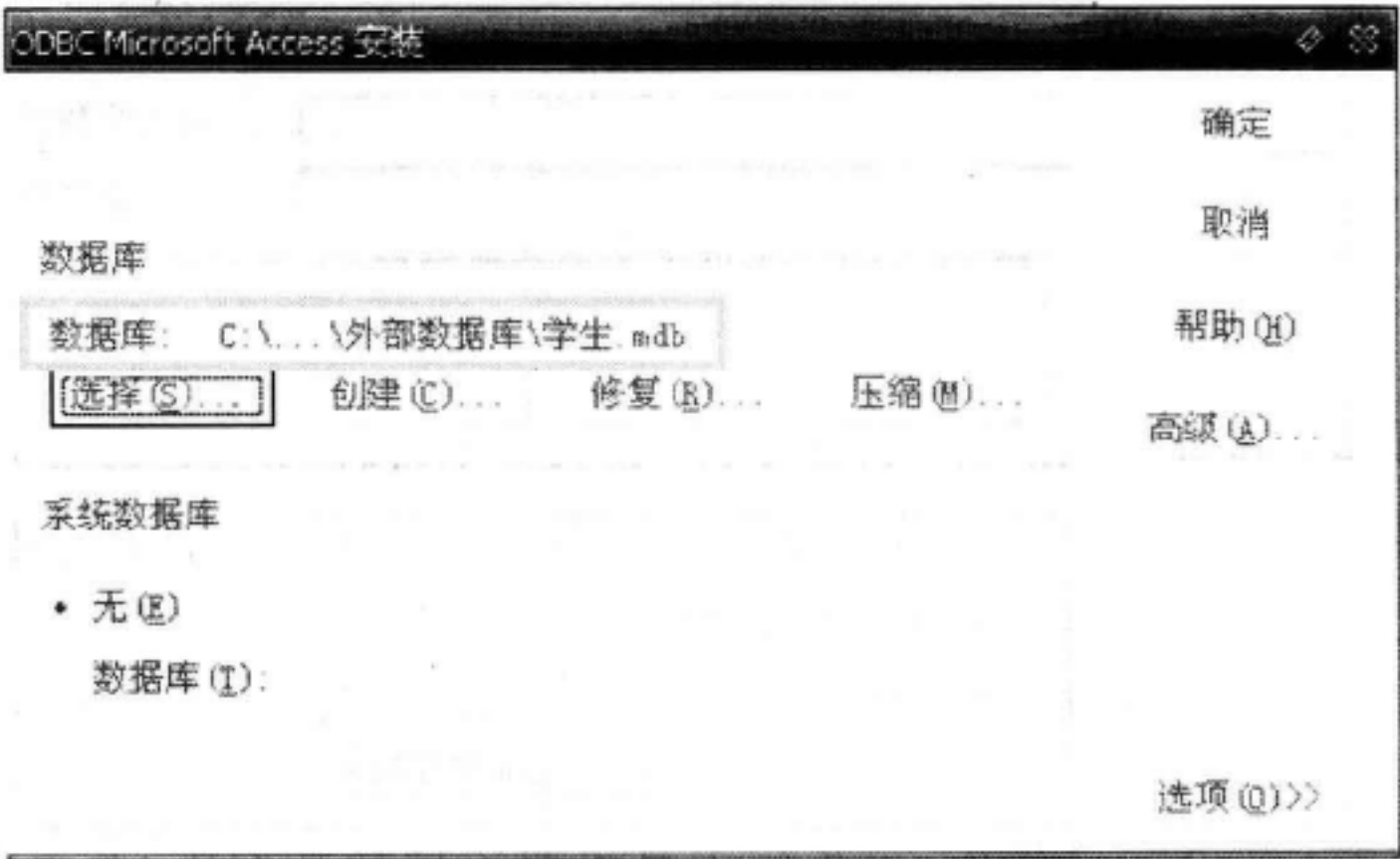


图 12-31 ODBC Microsoft Access 安装窗口

至此,用 ODBC 连接外部数据库的操作全部完成。这是 ODBC 配置的标准过程,完全由微软定义的过程。值得一提的是,下次再连接同一数据库时,就不必再新建一个 *.dsn 文件了,直接在最前面的对话框(如图 12-25 所示)中选择 *.dsn 就可以了。

【运行结果】运行例程 12-5,观测进行了外部数据库连接操作后,正确读出并显示 Access 数据库的第 1 条记录的效果,如图 12-32 所示。

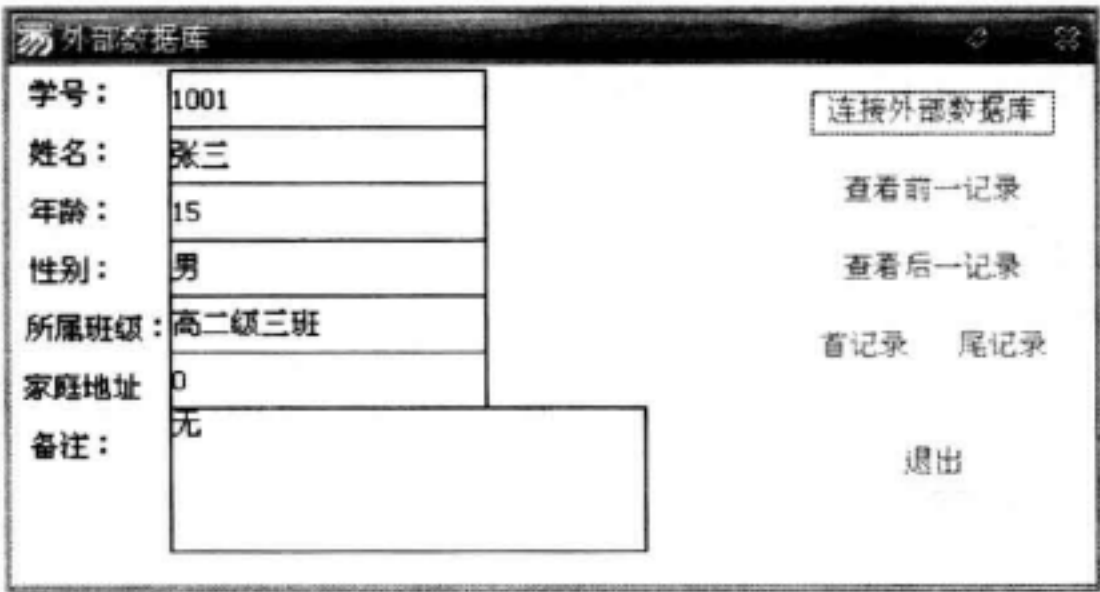


图 12-32 连接数据库之后的程序主界面

主要的程序代码如图 12-33 所示。

子程序名	返回值类型	公开	备注
_连接外部数据库按钮_被单击			

```
如果真 (外部数据库1. 打开 ( ))
    记录集句柄 = 外部数据库1. 查询 ("select * from student")
    如果 (记录集句柄 = 0)
        信息框 ("查询失败!", 0, )
    外部数据库1. 到首记录 (记录集句柄)
    读取并显示当前记录 0
```

子程序名	返回值类型	公开	备注
读取并显示当前记录			自定义子程序

```
编辑框1. 内容 = 到文本 (外部数据库1. 读 (记录集句柄, 1))
编辑框2. 内容 = 外部数据库1. 读 (记录集句柄, 2)
编辑框3. 内容 = 外部数据库1. 读 (记录集句柄, 3)
编辑框4. 内容 = 外部数据库1. 读 (记录集句柄, 4)
编辑框5. 内容 = 外部数据库1. 读 (记录集句柄, 5)
编辑框6. 内容 = 外部数据库1. 读 (记录集句柄, 6)
编辑框7. 内容 = 外部数据库1. 读 (记录集句柄, 7)
```

子程序名	返回值类型	公开	备注
_查看前一记录按钮_被单击			

```
外部数据库1. 到前一记录 (记录集句柄)
如果 (外部数据库1. 首记录前 (记录集句柄))
    信息框 ("前面已无记录!", 0, )
    外部数据库1. 到首记录 (记录集句柄)
读取并显示当前记录 0
```


子程序名	返回值类型	公开	备注
_首记录按钮_被单击			

```
外部数据库1. 到首记录 (记录集句柄)
读取并显示当前记录 0
```

图 12-33 启动窗口程序集


【范例解析】本例演示了在易语言中应用“外部数据库”组件,通过 ODBC 操纵 Microsoft Access 数据库的方法。外部数据库组件支持记录的查询、排序、删除、添加、更新及创建、删除表的操作。

12.5 外部数据提供者

外部数据提供者组件也是一个不可视组件,它通常通过数据源组件对外部数据提供者中的数据进行操作。

外部数据提供者组件除基本属性外,还有两个重要的属性:“连接文本”、“查询 SQL”。

1. “连接文本”属性

“连接文本”属性为文本型,用于设置外部数据库的 ODBC 连接文本。当在属性夹单击按钮将弹出 ODBC 数据源配置对话框。

2. “查询 SQL”属性

“查询 SQL”属性为文本型,用作指定数据库中的数据表名或者用作查询记录集的 SELECT 类 SQL 语句。

外部数据提供者组件没有专有方法,也没有专有事件。

【范例 12-6】用表格组件显示 Microsoft Access 数据库。在启动窗口上放置表格、数据源、外部数据提供者各一个,如图 12-34 所示。将表格 1 的“数据源”属性设置为“数据源 1”。

将数据源 1 的“数据提供者”属性设置为“外部数据提供者 1”。具体参考例程 12-6。

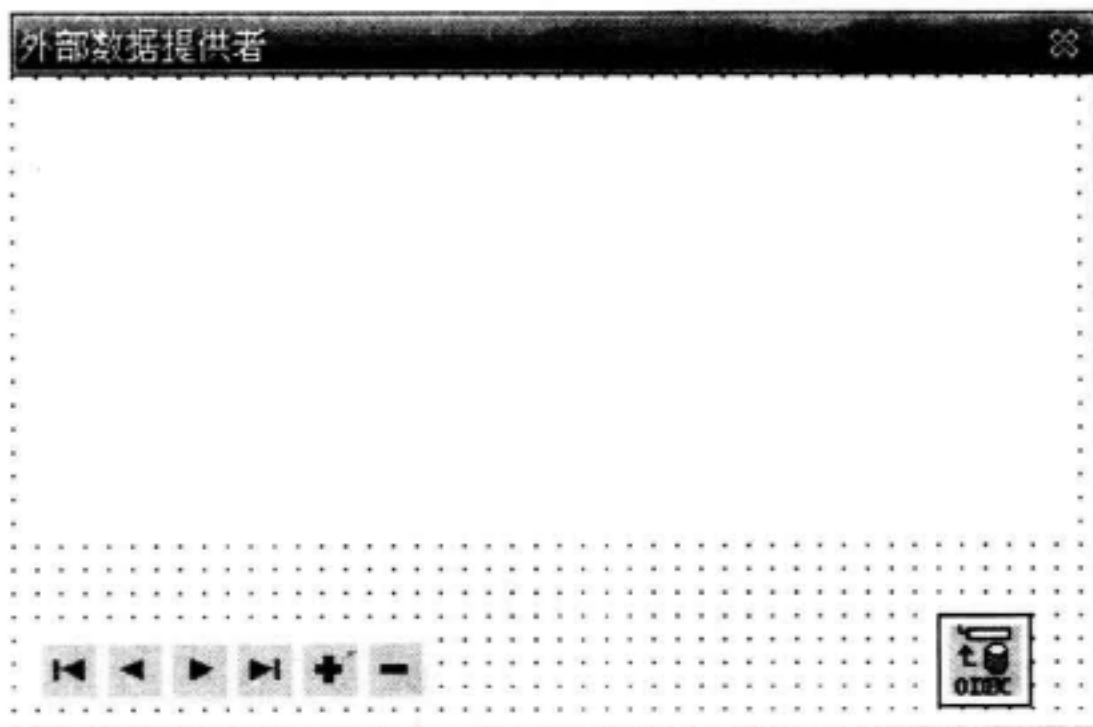


图 12-34 程序布局

单击外部数据提供者 1 的“连接文本”属性,将弹出配 ODBC 置对话框,新建或选择相应的 *.dsn 文件,具体过程前文中已经介绍过,此处不再赘述,如图 12-35 所示。

设置外部数据提供者 1 组件的“查询 SQL”属性为 `select * from books`。这里 books 是表名,依数据库而定。

【运行结果】运行例程 12-6,观测通过表格显示 Access 数据库中数据的效果,如图 12-36 所示。

【注意】数据源的删除记录按钮不可随便单击,该操作是不可逆的。

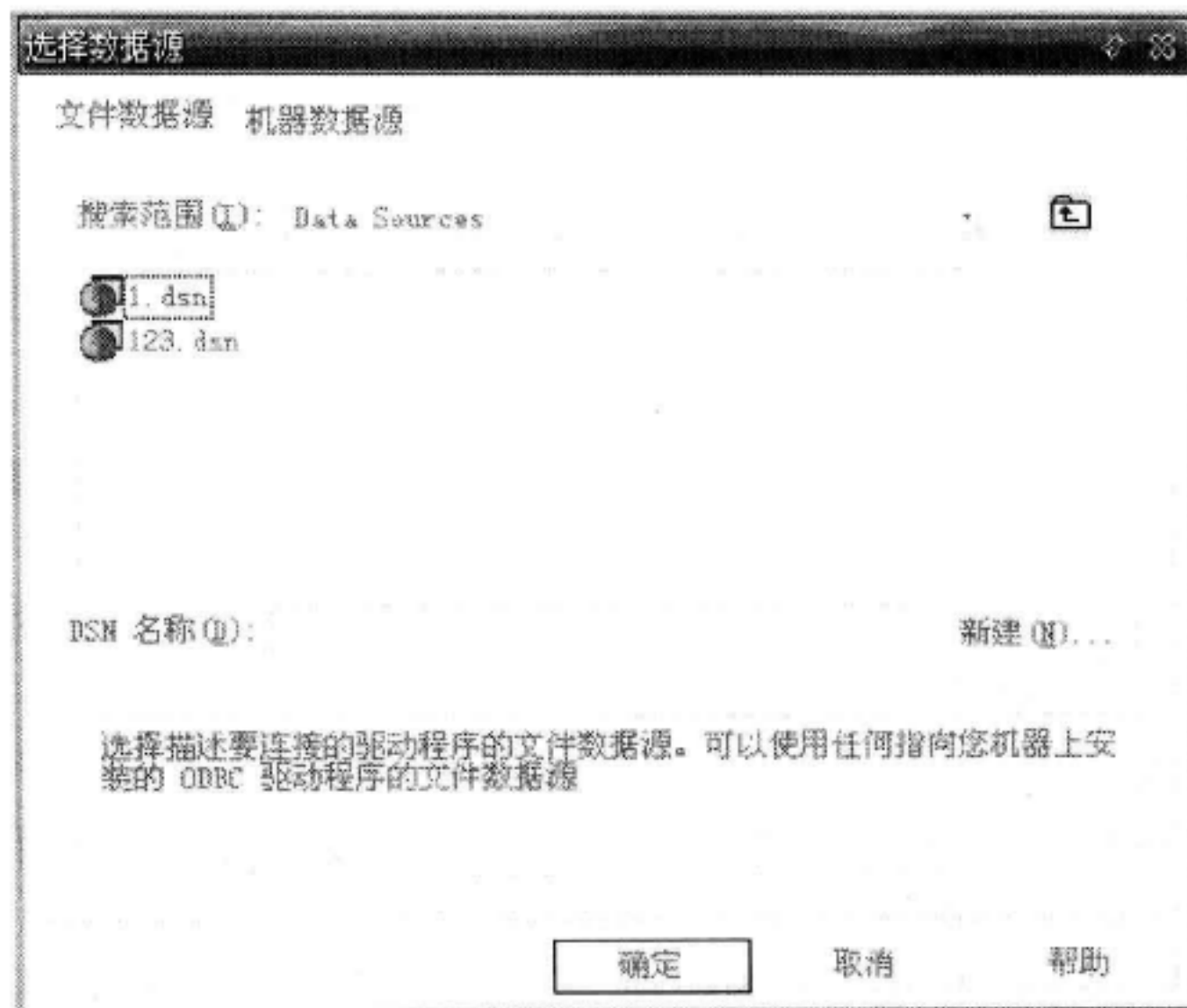


图 12-35 选择数据源对话框

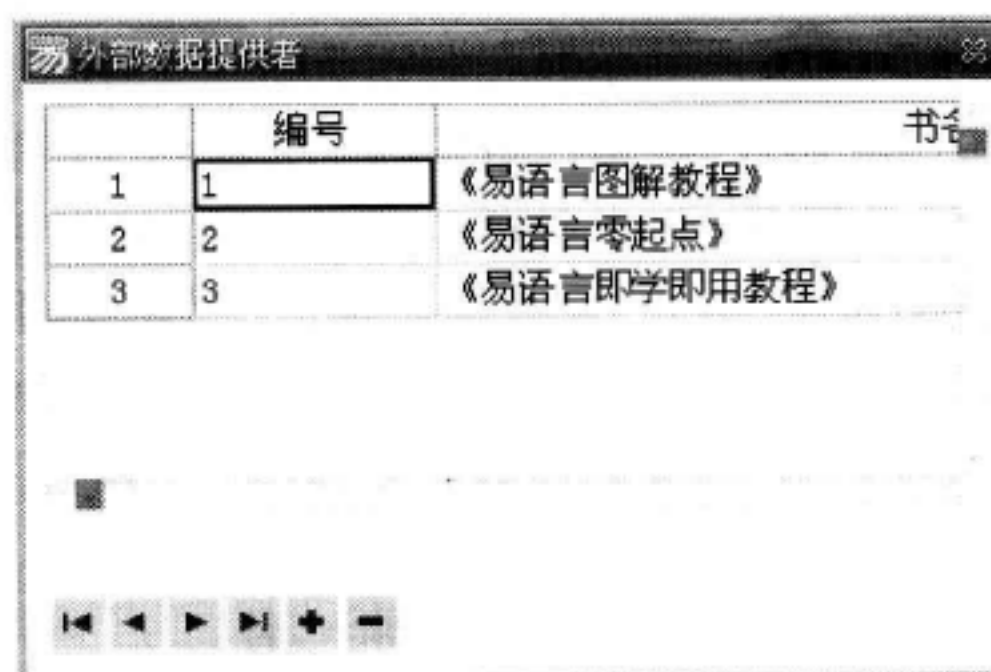


图 12-36 程序运行结果

第 13 章 网络组件

网络的建设促进了计算机技术的发展,特别是个人电脑的普及,让众多计算机用户能便捷地共享信息资源,同时使不同计算机之间也能互相传递信息进行交流。


通过网络交互数据,首先要明白“连接”和“无连接”的概念。最简单的例子莫过于打电话和写信。两个人如果要通电话,得首先建立连接,即拨号,等待对方应答后才能相互传递信息,最后还要释放连接,即挂电话。写信就没有那么复杂了,地址、姓名填好以后直接往邮筒一扔,收信人就能收到。用数据报组件进行数据交互,类似于写信,是基于“无连接”的;用客户、服务器进行数据交互,类似于打电话,是基于“连接”的。

根据提供服务类型的不同,端口分为两种,一种是 TCP 端口,一种是 UDP 端口。计算机之间相互通信的时候,分为两种方式:一种是发送以后就不管了,不去确认信息是否到达,这种方式大多采用 UDP 协议;一种是发送信息以后,可以确认信息是否到达,也就是有应答的方式,这种方式大多采用 TCP 协议。这两种交互方式分别类似于生活中的写信和打电话。在易语言中,用数据报组件封装了 UDP 协议,用客户组件和服务组件封装了 TCP 协议。

使用本章节介绍的组件,必须有一些其他的软硬件支持,如网卡、调制解调器等;还要正确安装了它们的软件。如使用数据报、客户、服务器组件前,要安装网卡、网卡驱动、协议等。如要进行通讯测试,还要有两台以上的电脑在局域网中,而且能在网上邻居中看到对方。若使用超级链接框与超文本浏览框组件,还要使用调制解调器上互联网进行测试。

13.1 数据报

13.1.1 数据报概述

数据报是一种不可靠、小数据量的网络数据交互方式。如果传递的数据量过大,有可能会失败,最好不要超过 127bit。如需要大数据量或者可靠数据传送方式,请使用面向连接的其他网络组件。数据报的优势在于无需建立连接,可批量群发,速度较快;劣势主要是不能保证发送的数据确实能到达目标。

13.1.2 数据报的重要属性

1. “端口”属性

“端口”属性是一个整数型的值,默认值是 19730。如果要改动,尽量取大于 1024 的较大值。

端口是很重要的概念。首先需要明白的一点是,我们这里所说的端口,不是计算机硬件的 I/O 端口,而是软件概念上的端口。

服务器可以向外提供多种服务,比如,一台服务器可以同时是 WEB 服务器,也可以是 FTP 服务器,同时,它也可以是邮件服务器。为什么一台服务器可以同时提供那么多的服务呢?其

中一个很主要的原因就是各种服务采用不同的“端口”来分别提供不同的服务。比如:HTTP(超文本传送)采用80端口;FTP(文件传输)采用21端口;Telnet(远程登录)采用23端口;POP3(邮件接收)采用110端口;SMTP(简单邮件传送)采用25端口;DNS(域名解析服务)采用53端口等。这样,通过不同端口,计算机与外界进行互不干扰的通信。

13.1.3 数据报的专有方法

1. “发送数据()”方法

功能:发送数据到指定主机上的指定端口。

语法:数据报名称.发送数据([接收主机地址],接收主机端口号,欲发送数据)

参数:接收主机地址——文本型,可以为主机名、IP地址等;如果省略本参数或者提供空文本,则在指定端口广播欲发送数据。接收主机端口号——整数型,必须是对方(接收主机)数据报组件的端口属性指定的数值。欲发送数据——欲发送的数据,可以是文本型、整数型、小数型、逻辑型、日期时间型等(数据类型不限)。

返回值:逻辑型。如果数据发送成功,返回“真”;如果发送失败,返回“假”。例如:

数据报.端口 = 2010

数据报.发送数据("127.0.0.1",2010,"大家好")

程序执行后将“数据报”所监听的端口设为“2010”;向本机的2010端口发送数据报,内容为“大家好”。

2. “取回数据()”方法

功能:取回数据报组件所接收到的数据。

语法:数据报名称.取回数据()

返回值:字节集型。

【注意】返回值必须是字节集型,使用时经常需要进行数据类型转换。例如:

收到的数据 = 数据报.取回数据()

输出调试文本(取字节集数据(收到的数据,#文本型,))

当“数据报”收到数据时,在输出面板中显示收到的数据。

13.1.4 数据报的重要事件

1. “数据到达”事件

本事件产生的时机是当有数据到达时自动产生。

在本事件的处理子程序中,一个最重要的任务就是取回“到达的数据”,通过“取回数据()”方法。还得再强调一次,“取回数据()”返回的是字节集型的数据,通常需要用数据类型转换函数“从字节集转换()”将字节集数据转换为其他数据类型(最常用的数据类型无疑是文本型)。

数据报的事件只有数据到达一个事件。例如:

编辑框1.加入文本(从字节集转换(数据报1.取回数据(),#文本型))

从上行代码可以看出,通过“数据报1.取回数据()”这个方法,将数据取到,并且转换为文本型数据,通过编辑框1进行显示。

【范例13-1】利用数据报编写一个简单的聊天工具,用于实现两台主机的互相发送信息。具体参考例程13-1。

详细的程序代码如图 13-1 所示。

窗口程序集名	保留	保留	备注
窗口程序集1			
变量名	类型	数组	备注
主机	文本型		

子程序名	返回值类型	公开	备注
_发送按钮_被单击			

如果 (数据报1.发送数据 (主机, 19730, 编辑框2.内容) = 真)
 编辑框1.起始选择位置 = -1
 编辑框2.内容 = ""
 编辑框2.获取焦点 0
 编辑框2.被选择字符数 = -1

子程序名	返回值类型	公开	备注
_连接按钮_被单击			

输入框 ("请输入欲连接到的主机地址 (可以为主机名、IP地址等)", , 取主机名 0, 主机, ,)
发送按钮.禁止 = 假

子程序名	返回值类型	公开	备注
_数据报1_数据到达			

变量名	类型	静态	数组	备注
收到文本	文本型			

收到文本 = 取字节集数据 (数据报1.取回数据 0, #文本型)

编辑框1.内容 = 编辑框1.内容 + 收到文本
编辑框1.内容 = 编辑框1.内容 + #换行符 + 取重复文本 (50, " ") + #换行符

图 13-1 数据报应用源代码

【运行结果】运行例程 13-1, 观测数据发送的过程, 如图 13-2 所示。

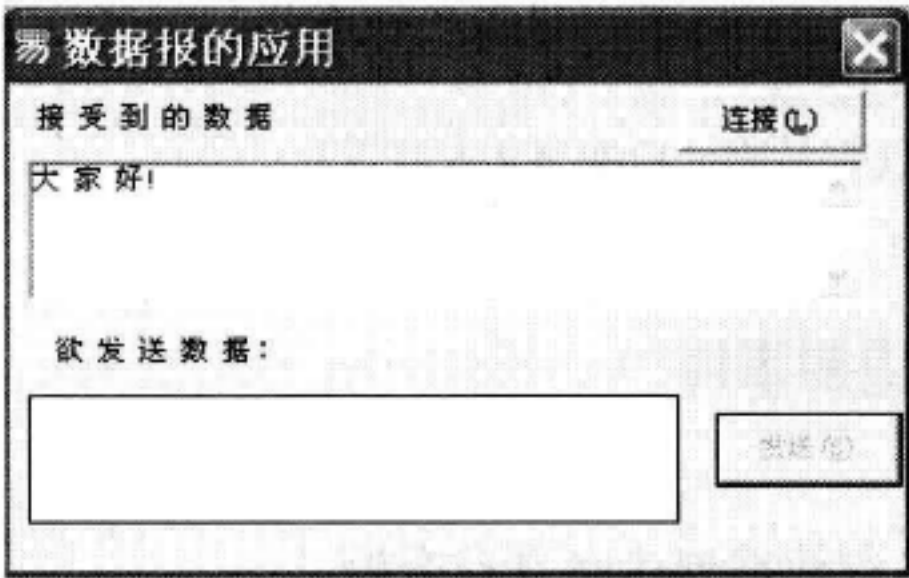




图 13-2 数据报应用运行结果

【注意】本程序只用到三个子程序, 其中连接按钮是取对方 (目的地方) 的主机名, 发送按钮是将编辑框中的文字发向对方。数据到达事件处理别人发过来的信息显示在编辑框中, 当然, 也可以自己发给自己进行测试。

13.2 客户/服务器组件

13.2.1 客户/服务器组件概述

客户组件  和服务端组件  是一对好搭档。它们总是成对使用, 分别处于两个独立的应

用程序中,一个充当客户端,一个充当服务器端。客户端的应用程序总是向服务器提出服务请求,而服务器端的应用程序则根据客户端的请求提供服务。可以说客户端是主动方,服务器端是被动方。当然也可以在一个应用程序中既使用客户组件,又使用服务器组件,这样它既充当客户端又充当服务器端。

13.2.2 客户组件的重要属性

客户组件只有“名称”、“备注”、“左边”、“顶边”、“宽度”、“高度”与“标记”基本属性,没有其他的属性,如图 13-3 所示。由于客户机组件在运行时是不可见组件,所以这些属性都可以不用设置。

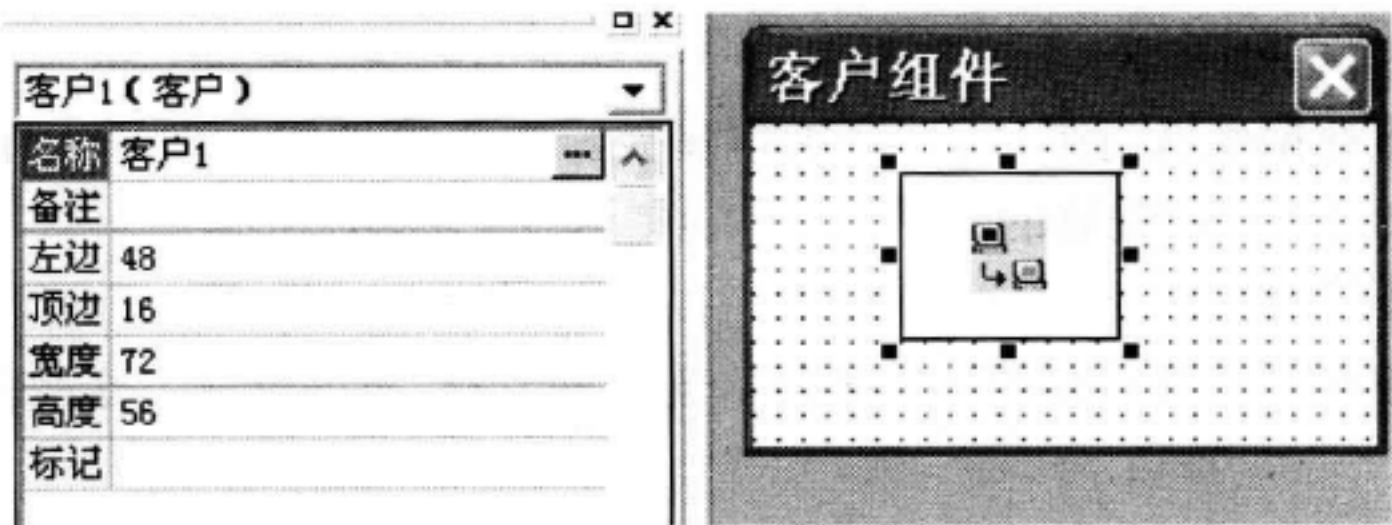


图 13-3 客户组件的重要属性

13.2.3 客户组件的专有方法

1. “连接()”方法

功能:连接到指定主机上的指定端口。

语法:客户名称. 连接(服务器地址,服务器端口号)

参数:服务器地址——文本型,可以为主机名、IP 地址;服务器端口号——整数型,即服务器组件的端口属性指定的值。例如:

客户 1. 连接(110.0.0.1,19730)

该实例表示与 IP 地址为 110.0.0.1 的主机中的服务器组件进行连接。该服务器组件的属性必须是 19730,且它所在的程序正在运行,否则连接不可能成功。

【注意】同一时刻,一个客户组件只能连接一个服务器组件;一个服务器组件可以同时跟多个客户组件连接。

2. “断开连接()”方法

功能:断开与服务器的连接。

语法:客户名称. 断开连接() 例如:

客户 1. 断开连接()

3. “发送数据()”方法

功能:向已经建立连接的服务器组件发送数据。

语法:客户名称. 发送数据(数据),参数可以是各种类型的数据。例如:

客户 1. 发送数据(0)

客户 1. 发送数据(编辑框 1. 内容)

客户 1. 发送数据(图片框 1. 图片)

4. “取回数据()”方法

功能:取回服务器发送来的数据。

语法:客户名称. 取回数据()

本方法返回字节集型的数据。使用时有可能需要进行数据类型转换(用“从字节集转换()”函数),这跟数据报组件的“取回数据”方法是完全一致的。

13.2.4 客户组件的重要事件

1. “数据到达”事件

当服务器端将数据发送过来后,会产生本事件。在本事件的处理子程序中调用“取回数据()”方法即可取回本次所收到的数据。

2. “连接断开”事件

当连接被服务器端断开后会产生本事件。连接断开后,不能继续发送数据,除非重新建立连接。

13.2.5 服务器的重要属性

“端口”属性:

整数型。指定监听数据到达的端口号,该端口号是大于 0 且小于 32767 的任何自定义数值。(应尽量取大于 1024 的较大值。)其含义与数据报组件的同名属性类似。

服务器与客户的端口号必须一致,否则不能相互通信。

13.2.6 服务器的专有方法

1. “取回客户()”方法

功能:取回与服务器连接的客户地址。

语法:服务器名称. 取回客户()

本方法返回一个文本型的值,其中记录了客户的地址(IP 地址 + 端口)。当服务器组件向客户发送数据或断开客户时,都需要指定该地址。例如:

客户地址 = 服务器 1. 取回客户

取回客户的地址,并保存到文本型变量“客户地址”中。“客户地址”要事先定义为全局变量或程序集变量,以供其他子程序使用。

【注意】通常在服务器组件的“客户进入”或“客户离开”(特别是“客户进入”)事件的处理子程序中调用本方法。

2. “取回数据()”方法

功能:取回客户发送来的数据。

语法:服务器名称. 取回数据()

返回值为字节集型。

3. “发送数据()”方法

功能:向指定客户发送数据。

语法:服务器名称. 发送数据(接收客户,数据,[最长等待时间])

参数:客户地址——文本型,必须是调用“取回客户”方法获得的客户地址。数据——可以是各种类型的数据。最长等待时间——指定等待发送成功的最长时间,单位为秒。如果省略本参数,默认为无限等待。例如:

服务器 1. 发送数据(客户地址,123,)

向发送整数型数据 123。这里的“客户地址”就是前面调用“取回客户()”时的返回值。

4. “断开客户()”方法

功能:与指定客户断开连接。

语法:服务器名称. 断开客户(欲断开客户)

参数必须是调用“取回客户()”方法所返回的客户地址文本。例如:

服务器 1. 断开客户(客户地址)

与“客户地址”所指定的客户断开连接。这里的“客户地址”就是前面调用“取回客户()”时的返回值。

13.2.7 服务器的重要事件

1. “数据到达”事件

当服务器端将数据发送过来后产生本事件。在本事件的处理子程序中调用“取回数据()”方法即可取回本次所收到的数据。

2. “客户进入”事件

当有新客户连接入本服务器组件后产生本事件。本事件的事件处理子程序的一个重要的任务就是:调用“取回客户()”方法获得新客户的地址,并保存到文本型的全局变量或程序集变量中,供以后使用(服务器组件的方法“发送数据()”“断开客户()”都需要指定客户的地址)。


3. “客户离开”事件

当有已连接客户断开与本服务器组件的连接后,会产生本事件。在本事件的处理子程序中调用“取回客户”方法即可取回此客户的地址(IP 地址+端口)。

本节中引用的代码参见例程 13-2。

13.3 超级链接框

13.3.1 超级链接框概述

通过超级链接框组件 , 可以从程序中链接到英特网或本机中的其他程序。因此,如果要应用超级链接框组件访问英特网,就必须要有上网的工具,如网卡、调制解调器,还要安装相关的驱动与协议等。

超级链接框的重要属性有:“标题”、“类型”、“电子信箱地址”、“Internet 地址”、“文本颜色”、“热点颜色”、“访问后的颜色”,此外还有“背景颜色”、“字体”等。

超级链接框的专有方法有:“跳转()”方法。

超级链接框只有 6 个基本事件,没有其他重要事件。

13.3.2 超级链接框的重要属性

1. “标题”属性

“标题”属性用于设置超级链接框中所显示的文字,属性类型是文本型。如果未指定标题文本,则默认以“电子信箱地址”属性或“Internet 地址”属性文本作为标题文本。如图 13 - 4 所示,未指定超级链接框标题文本,指定其 Internet 地址为: `http://www.eyuyan.com`,此时就以 Internet 地址 `http://www.eyuyan.com` 作为标题文本。

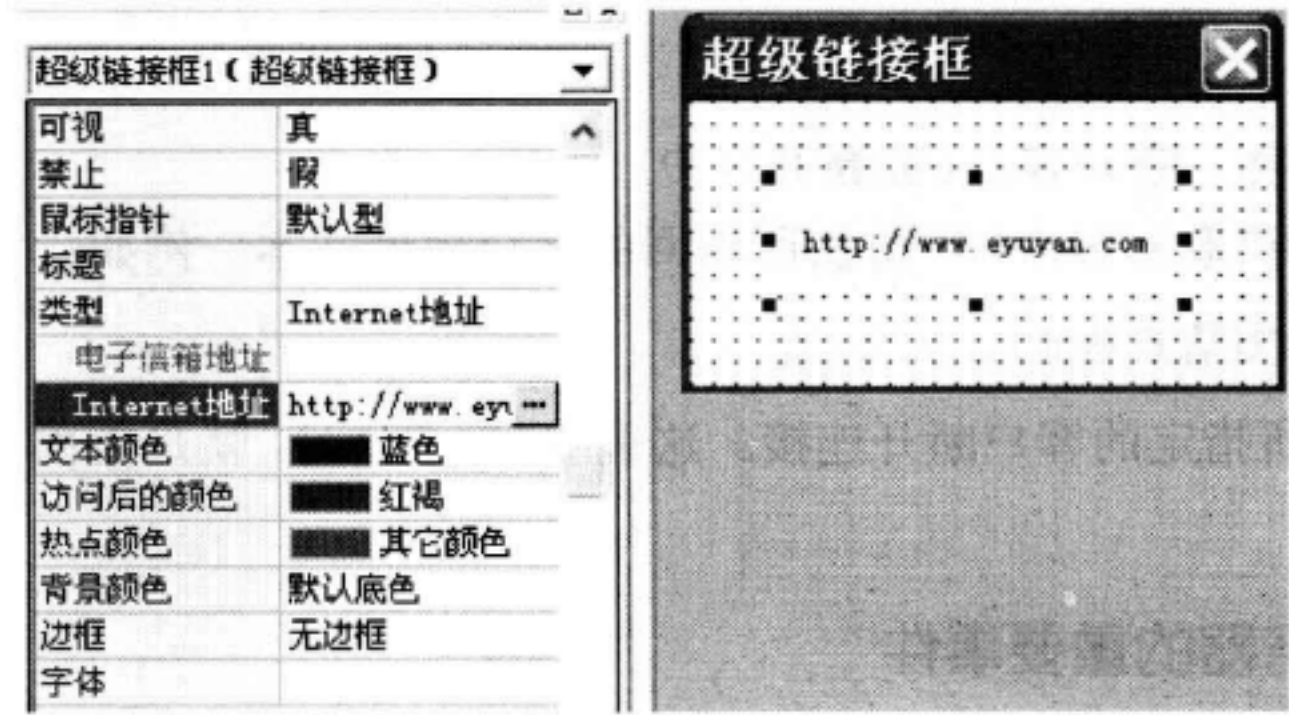


图 13 - 4 超级链接框标题属性

事实上,“电子信箱地址”属性或“Internet 地址”属性文本可以与标题不一致。如图 13 - 5 所示,超级列表框标题属性为“易语言”,Internet 地址为: `http://www.eyuyan.com`,运行后该超级列表框仍显示为标题属性“易语言”,单击运行效果与图 13 - 5 一样,都是链接到目标地址为: `http://www.eyuyan.com` 的网站。

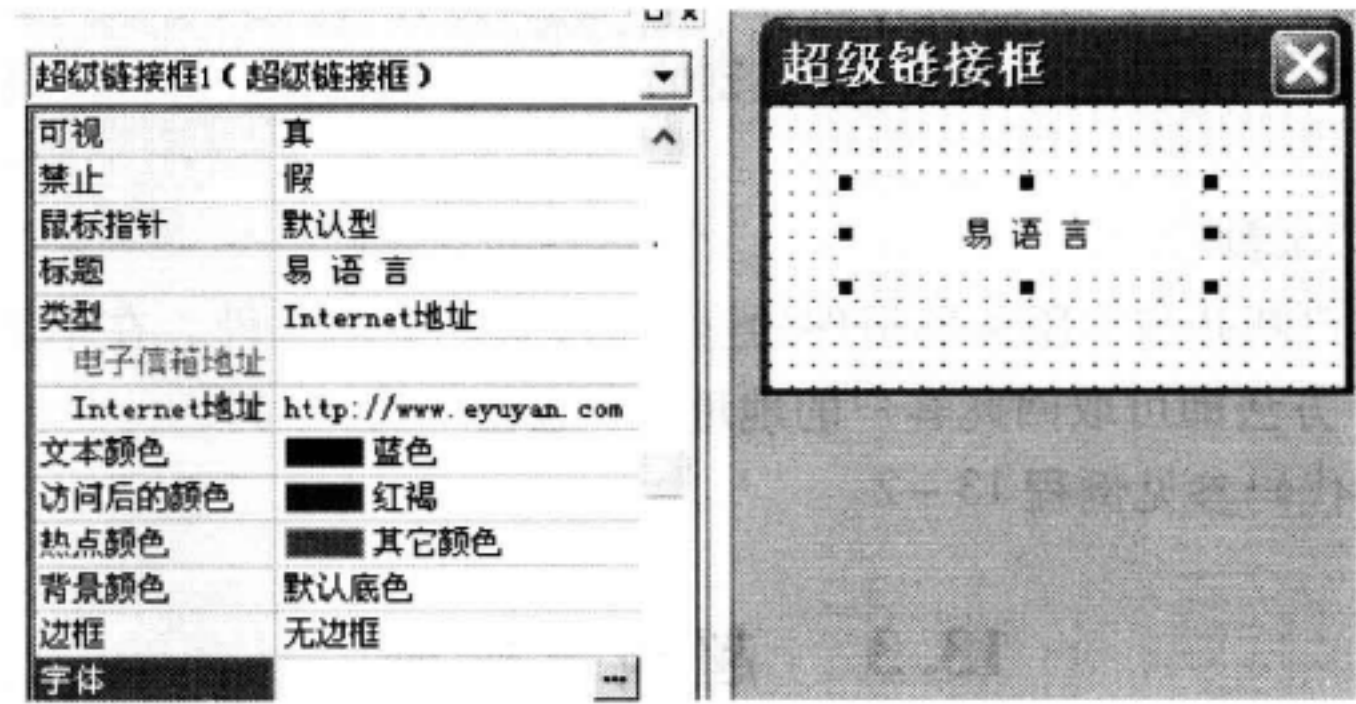


图 13 - 5 超级列表框标题属性

也可以通过下面这个命令行在程序中将标题属性改变:

```
超级链接框_标题.标题 = “易语言”
```

2. “类型”属性

本属性用于指定超级链接框的链接类型,有两个可选值:0. 电子信箱地址、1. Internet 地址,默认为 0。当类型 = 0 时,超级链接框的另外一个属性“电子信箱地址”属性有效;当类型 = 1 时,“Internet 地址”属性有效。例如:

```
超级列链接框_类型.类型 = 1
```


程序执行后设置“超级链接框_类型”的链接类型为 Internet 地址。

3. “电子信箱地址”属性

如果“类型”属性值为“0”时。本属性用于指定链接到的目标邮箱地址。程序运行后,点击标题后会自动起用电子邮箱软件,并填写发信目标未属性指定的地址。例如:

超级链接框_地址. 电子信箱地址 = “eyuyan@ sina. com. cn ”

程序执行后设置“超级链接框_地址”的电子信箱目标地址为“eyuyan@ sina. com. cn”。

4. “Internet 地址”属性

如果“类型”属性值为“1”时,本属性用于指定链接到的目标 Internet 地址。程序运行后,点击标题后会自动跳转到本属性所指定的地址。例如:

超级链接框_地址. Internet 地址 = “http://www. eyuyan. com”

程序执行后将“超级链接框_地址”的 Internet 链接目标地址为:“http://www. eyuyan. com”。

5. “文本颜色属性”

本属性用于指定正常情况下(未点击)时的标题文本显示颜色,默认为“蓝色”。属性值为 RGB 颜色值,可以通过“取颜色值”命令调配,或使用易语言中内置的颜色常量,例如:

超级链接框 1. 文本颜色 = #蓝色

表示未访问时文本颜色为蓝色。

6. “访问后的颜色”属性

本属性用于指定被访问过后的标题文本显示颜色。属性值为 RGB 颜色值,可以通过“取颜色值”命令调配,或使用易语言中内置的颜色常量。例如:

超级链接框 1. 访问后的颜色 = #红褐

程序执行完,点击访问后文本颜色为红色。

7. “热点颜色”属性

本属性用于指定鼠标移动到组件上时的文本显示颜色。属性值为 RGB 颜色值,可以通过“取颜色值”命令调配,或使用易语言中内置的颜色常量,例如:

超级链接框 1. 热点颜色 = #黄色

程序执行后,鼠标移到标题上时标题的热点颜色为黄色。

8. “背景颜色”属性

本属性用于指定超级链接框的背景显示颜色。属性值为 RGB 颜色值,可以通过“取颜色值”命令调配,或使用易语言中内置的颜色常量,例如:

超级链接框 1. 背景颜色 = #灰色

程序执行后,整个超级列表框的背景颜色为灰色。

9. “边框”属性

本属性用于指定超级链接框的显示样式。可供选择的属性值有:0(无边框)、1(凹入式)、2(凸入式)、3(浅凹入式)、4(镜框式)、5(单线边框式)。例如:

超级链接框_边框. 边框 = 5

程序执行后将“超级链接框_边框”的边框设为单线边框样式。

13.3.3 超级链接框的专有方法

超级链接框的专有方法为“跳转()”方法,主要是跳转到指定的邮件或 Internet 地址。当

超级链接框被鼠标单击时,它会自动跳转到相应地址,不必显示调用“跳转()”方法。例如:

超级链接框 1. 类型 = 1

超级链接框 1. Internet 地址 = <http://www.eyuyan.com>

超级链接框 1. 跳转()

程序执行后,自动跳转到 <http://www.eyuyan.com>

【范例 13-3】利用超级列表框实现单击一张图片也可以进入一个网站。具体参考例程 13-3。

方法一:通过点击两张不同的图片进入两个不同的网站,这种实现方式有两种,第1种方法即是使用两个叠加的组件——图片框与超级链接框,都为可视属性,超级链接框比图片框大2个像素,即可实现单击后进入相关网站的功能。并且鼠标状态是手形状态。如图13-6所示设置超级链接框的属性。程序运行后,点击图片框1即跳到<http://www.sina.com.cn>。



图 13-6 超级链接框 1 属性设置

第2种方法第2种方法即是使用跳转的方法,在此还使用“鼠标指针 = 8. 十字型”,来让鼠标有所区别,具体的程序代码如图13-7所示。

窗口程序集名	保 留	保 留	备 注
窗口程序集1			

子程序名	返回值类型	公开	备 注		
_图片框2_鼠标左键被按下	逻辑型				
参数名	类 型	参考	可空	数组	备 注
横向位置	整数型				
纵向位置	整数型				
功能键状态	整数型				

超级链接框1.Internet地址 = "http://www.dywt.com.cn"

超級链接框1. 跳转 (0)

子程序名	返回值类型	公开	备 注		
_图片框2_鼠标位置被移动	逻辑型				
参数名	类 型	参 考	可 空	数 组	备 注
横向位置	整数型				
纵向位置	整数型				
功能键状态	整数型				

图片框1.鼠标指针 = { 8 }

图 13-7 超级链接框应用运行代码


【运行结果】运行例程 13-3, 观测单击图片后的变化, 如图 13-8 所示。



图 13-8 超级链接框运行结果

13.4 超文本浏览框组件

13.4.1 超文本浏览框组件概述

利用超文本浏览框组件 , 可实现对浏览器的编写。超文本浏览框组件的重要属性有: “离线浏览”、“静默”、“地址”、“状态条文本”、“标题”、“字体大小”等。

超文本浏览框的重要事件有: “即将跳转”、“跳转完毕”事件、“载入开始”事件、“载入进度改变”事件、“载入完毕”事件、“已就绪”事件、“状态文本被改变”事件、“标题被改变”事件、“命令状态被改变”、“即将打开新窗口”事件等。

超文本浏览框的重要方法有: “执行命令()”方法、“跳转()”方法、“取文档类型()”方法、“是否就绪()”方法等。

13.4.2 超文本浏览框组件的重要属性

1. “离线浏览”属性

该属性指定浏览器是否从缓存里面读取超文本页面数据。属性类型为逻辑型。如果为“真”, 以前浏览的网页时会有临时文件存放在系统中, 这时会读取这些文件并显示出来。如果为“假”, 那么将会从 Internet 上下载文件显示, 这时显示的是真正即时的网上信息。

例如:

超文本浏览框 1. 离线浏览 = 真

程序执行完后, 浏览器能够从缓存里面读取超文本页面数据。

2. “静默”属性

其属性类型也是逻辑型, 如果为“真”, 则浏览过程中不允许浏览器显示自己的对话框。如果为“假”, 则浏览过程中允许浏览器显示自己的对话框。例如:

超文本浏览框 1. 静默 = 真

程序执行完后, 浏览过程中不允许浏览器显示自己的对话框。

3. “地址”属性

该属性指定欲显示的网络或本机超文本页面地址, 其类型值是文本型。例如:

超文本浏览框 1. 地址 = “http://www. eyuyan. com”

程序执行完后,显示网络超文本页面地址是 http://www. eyuyan. com 。

4. “状态条文本”属性

用作在运行时提供浏览器的当前状态条文本。其类型为文本型,可提供给状态条组件以显示。例如:

状态条 1. 置文本(1,超文本浏览框 1. 状态条文本)

程序执行完后,将超文本浏览框的状态条文本提供给状态条组件。

5. “标题”属性

用作在运行时提供浏览器的当前标题文本,其类型是文本型,一般在网页被单击改变后会更换为新网页的标题,这时就可以用这个属性取到新标题,并且刷新显示新标题到提示信息中。例如:

_启动窗口. 标题 = 超文本浏览框. 标题

程序执行后,启动窗口的标题更换为超文本浏览框的标题。

6. “字体大小”属性

一共有 5 种类型可以选择:“0. 最小”、“1. 较小”、“2. 中等”、“3. 较大”、“4. 最大”。默认是“2. 中等”。例如:

超文本浏览框. 标题 = 4

程序执行后,超文本浏览框的字体为最大。

13.4.3 超文本浏览框组件的重要事件

1. “即将跳转事件”

在浏览器即将跳转到另一个页面之前产生此事件,在事件处理子程序中读取“地址”属性即可得知即将跳转到的地址,返回“假”不允许跳转,返回“真”或不返回值允许跳转。

例如:

网页状态条. 置文本(0,“超文本浏览框……. 即将跳转”)

程序执行完后,浏览器即将跳转到另一页面时,网页的状态条显示“超文本浏览框... 即将跳转”。

【范例 13-4】用超文本浏览框组件编写一个浏览器,当浏览器开始跳转时触发“即将跳转事件”,同时在网页状态条显示“超文本浏览框即将跳转”。具体参考例程 13-4。

程序代码如图 13-9 所示。

【运行结果】运行例程 13-4,观测单击按钮后超文本浏览框的变化,如图 13-10 所示。

【注意】当点击跳转按钮时触发“即将跳转事件”,同时网页状态条显示“超文本浏览框... 即将跳转”该事件只有在跳转时有效,用地址属性打开网页该事件无效。

2. “跳转完毕”事件

当浏览器已跳转到另一个页面之后产生此事件,在事件处理子程序中读取“地址”属性即可得知已跳转到的地址。如图 13-11 所示。

程序执行完后,浏览器已跳转到另一个页面后,网页状态条显示“超文本浏览框... 跳转完毕”信息。

窗口程序集名	保留	保留	备注
启动窗口程序集			

子程序名	返回值类型	公开	备注
_跳转按钮_被单击			

超文本浏览框.跳转 (网址编辑框.内容, ,)			
子程序名	返回值类型	公开	备注
_超文本浏览框_即将跳转	逻辑型		
网页状态条.置文本 (0, “超文本浏览框.....即将跳转”)			
子程序名	返回值类型	公开	备注
_超文本浏览框_跳转完毕			
网页状态条.置文本 (0, “超文本浏览框.....跳转完毕”)			

图 13-9 超文本浏览器程序代码



图 13-10 超文本浏览器运行结果

3. “载入开始”事件

本事件在“即将跳转”事件之后触发,表示浏览器已开始载入将要显示的文档。如图 13-12 所示。

子程序名	返回值类型	公开	备注
_超文本浏览框_跳转完毕			

网页状态条.置文本 (0, “超文本浏览框.....跳转完毕”)

图 13-11 “跳转完毕”事件

子程序名	返回值类型	公开	备注
_超文本浏览框_载入开始			

网页状态条.置文本 (0, “载入开始”)

图 13-12 “载入开始”事件

程序执行完后,“即将跳转”事件触发后,网页状态条显示“载入开始”信息。

4. “载入进度改变”事件

在浏览器载入文档的过程中,每当文档被载入一部分即触发本事件,用作通知载入进度。如图 13-13 所示。

程序执行完后,网页状态条显示“载入进度改变”信息,网页进度条显示文档载入进度。

5. “载入完毕”事件

当将要显示在浏览器内的文档被载入完毕后触发本事件。如图 13-14 所示。

程序执行完后,浏览器内的文档被载入完毕后,网页状态条显示“载入完毕”信息。

子程序名	返回值类型	公开	备 注		
_超文本浏览框_载入进度改变					
参数名	类 型	参 考	可 空	数 组	备 注
进度百分比	整数型				
网页状态条.置文本 (1, “载入进度改变”)					
网页进度条.位置 = 进度百分比					

图 13-13 “载入开始”事件

6. “已就绪”事件

当浏览器已经将所需显示的文档处理完毕后发送本事件,在事件处理子程序中读取“地址”属性即可得知已就绪文档的地址。如图 13-15 所示。

子程序名	返回值类型	公开	备 注
_超文本浏览框_载入完毕			
网页状态条.置文本 (2, “载入完毕”)			

图 13-14 “载入完毕”事件

子程序名	返回值类型	公开	备 注
_超文本浏览框_已就绪			
网页状态条.置文本 (3, “已就绪”)			

图 13-15 “已就绪”事件

程序执行完后,浏览器处理完所需文档后,网页状态条显示“已就绪”信息。

7. “状态文本被改变”事件

当浏览器的状态条文本被改变后发送本事件,在事件处理子程序中读取“状态条文本”属性即可得知其内容。如图 13-16 所示。

8. “标题被改变”事件

当浏览器的标题文本被改变后发送本事件,在事件处理子程序中读取“标题”属性即可得知其内容。如图 13-17 所示。

子程序名	返回值类型	公开	备 注
_超文本浏览框_状态文本被改变			
网页状态条.置文本 (0, 超文本浏览框.状态条文本)			

图 13-16 “状态文本改变”事件

子程序名	返回值类型	公开	备 注
_超文本浏览框_标题被改变			
_启动窗口.标题 = 超文本浏览框.标题			

图 13-17 “标题被改变”事件

程序执行完后,当超文本浏览框的标题被改变后,启动窗口的标题显示当前超文本浏览框的标题。

9. “命令状态被改变”事件

当“前进”、“后退”等命令的允许状态被改变后发送此事件,用户程序应该根据状态值允许或禁止对应的按钮或菜单项。如图 13-18 所示。

子程序名	返回值类型	公开	备 注		
_超文本浏览框_命令状态被改变					
参数名	类 型	参 考	可 空	数 组	备 注
命令	整数型				
是否被允许	逻辑型				
判断 (命令 = #前进)					
网页状态条.置文本 (1, “前进到下一页”)					
判断 (命令 = #后退)					
网页状态条.置文本 (1, “后退到上一页”)					

图 13-18 “命令状态被改变”事件

程序执行完后,当命令状态变为“前进”命令时,网页状态条显示“前进到下一页”。

当命令状态变为“后退”命令时,网页状态条显示“后退到上一页”。

10. “即将打开新窗口”事件

在浏览器即将打开新窗口浏览另一个页面之前产生此事件,事件处理子程序返回“假”表示不允许打开,返回“真”或不返回值则允许打开。

13.4.4 超文本浏览框组件的重要方法

1. “执行命令()”方法

执行指定的浏览器命令。

调用格式:〈无返回值〉对象. 执行命令(欲执行的命令)。其中“欲执行的命令”,欲执行的命令。如图 13 - 19 所示。

子程序名	返回值类型	公开	备注
_后退超级按钮_被单击			
超文本浏览框. 执行命令 (#后退)			
子程序名	返回值类型	公开	备注
_前进超级按钮_被单击			
超文本浏览框. 执行命令 (#前进)			

图 13 - 19 “执行命令”方法

程序执行后,当点击后退超级按钮时,超文本浏览框就自行向后退,点击前进超级按钮时,超文本浏览框就自行前进。

2. “跳转()”方法

打开互联网或者本机上指定位置处的文档,本命令为初级对象成员命令。

调用格式:〈无返回值〉对象. 跳转(欲打开文档地址,[CHM 文件名称],[目标窗口名称])

“欲打开文档地址”,类型为“文本型(text)”。指定文档位于互联网或本机上的地址。

“CHM 文件名称”,类型为“文本型(text)”,可以被省略。如果欲打开文档处于 .CHM 文件内,本参数提供该 CHM 文件的名称。如果本参数被省略,默认值为空文本。

“目标窗口名称”,类型为“文本型(text)”,可以被省略。指定文档是否在单独窗口中打开,相同的名称指定相同的窗口。如果欲在浏览框内部打开,提供空文本参数值即可。如果本参数被省略,默认值为空文本。如图 13 - 20 所示。

程序执行完后,单击跳转超级按钮后,超文本浏览框就跳转到网址编辑框当前的网址。

3. “取文档类型()”方法

返回超文本浏览框中现行文档的类型文本。本命令为初级对象成员命令。如图 13 - 21 所示。

子程序名	返回值类型	公开	备注
_跳转超级按钮_被单击			

超文本浏览框. 跳转 (网址编辑框. 内容, ,)

图 13 - 20 “跳转”方法

子程序名	返回值类型	公开	备注
_超文本浏览框_载入完毕			

网页状态条. 置文本 (1, 超文本浏览框. 取文档类型 ())

图 13 - 21 “取文档类型”方法

程序执行完后,超文本浏览框载入完毕后,网页状态条显示当前文档或网页的内容。

4. “是否正在下载()”方法

如果当前正在下载文档或执行页面跳转,返回“真”,否则返回“假”。本命令为初级对象

成员命令。

调用格式:〈逻辑型〉对象.是否正在下载()。如图 13-22 所示。

子程序名	返回值类型	公开	备注
_超文本浏览框_载入完毕			

网页状态条.置文本 (2, 选择 (超文本浏览框.是否正在下载 0, “下载中”, “载入完毕”))

图 13-22 “是否正在下载”方法

程序执行完后,超文本浏览框载入完毕后,如果当前正在下载文档,网页状态条显示“下载中信息”,否则显示“载入完毕”信息。

5. “是否就绪()”方法

当浏览框已经准备时,返回“真”,否则返回“假”。本命令为初级对象成员命令。

调用格式:〈逻辑型〉对象.是否就绪()。如图 13-23 所示。

子程序名	返回值类型	公开	备注
_超文本浏览框_载入完毕			

网页状态条.置文本 (3, 选择 (超文本浏览框.是否就绪 0, “已就绪”, “未就绪”))

图 13-23 “是否就绪”方法

程序执行完后,超文本浏览框载入完毕后,查看超文本浏览框是否就绪,如果就绪网页状态条显示“已就绪”信息,否则显示“未就绪”。

13.5 远程服务支持库

13.5.1 远程服务支持库概述

本支持库提供对服务器端程序和客户端程序的支持。客户端发出请求,服务器端接收到客户端的请求后,就会执行事先设置好的触发函数,在触发函数的方法中就可以分析用户端的请求,然后计算出结果发回给相应的客户端。

远程服务支持库提供了类似“客户/服务器”组件的功能。该支持库是采用调用对象的形式使用的,因为它并不是组件,所以服务器端和客户端的功能都是通过对象的方法实现的。本支持库提供了 2 种数据类型,包括“远程服务”和“请求客户端”,分别来实现服务器端和客户端的功能。本支持库的基本使用方法为:在服务端程序和客户端程序中分别创建服务端和客户端变量,将服务端和客户端变量的数据类型设置为本支持库提供的“远程服务”和“请求客户端”数据类型,然后便可以使用相应对象提供的方法了。

13.5.2 远程服务的方法

1. “启动()”方法

服务参数配置完毕后,即可以启动服务。在使用服务器前,首先要使用此方法将服务器端启动。该方法执行成功返回“真”,执行失败返回“假”。

调用格式:〈逻辑型〉远程服务.启动(端口号,处理函数,[处理方式])

其中,端口号,整数型。本参数作为服务启动的端口号。当客户端与服务器端连接时,必

须要填写和服务端相同的端口号。

处理函数,子程序指针。本参数作为服务器端接收到客户请求后,系统自动调用的函数。在该函数内可以得到客户发送过来的信息。该函数带有一个整型参数,参数的内容是客户端发送过来的信息地址,可以通过该地址得到信息的完整内容。

处理方式,逻辑型。本参数作为服务程序的处理函数是否可以并行处理每一个客户端的请求。该参数为真代表处理函数可以并行处理每一个客户端的请求,这种方式下,服务器的性能可以最好的发挥出来,最大可能的支持更多的客户端访问和最快速的响应,这是默认的方式。该参数为假代表处理函数必须串行处理每一个客户端的请求,这种方式下,服务器的性能受到一定的限制,优点是不会发生由于客户端对服务器的并发访问而引起的并发冲突。如图 13-24 所示。

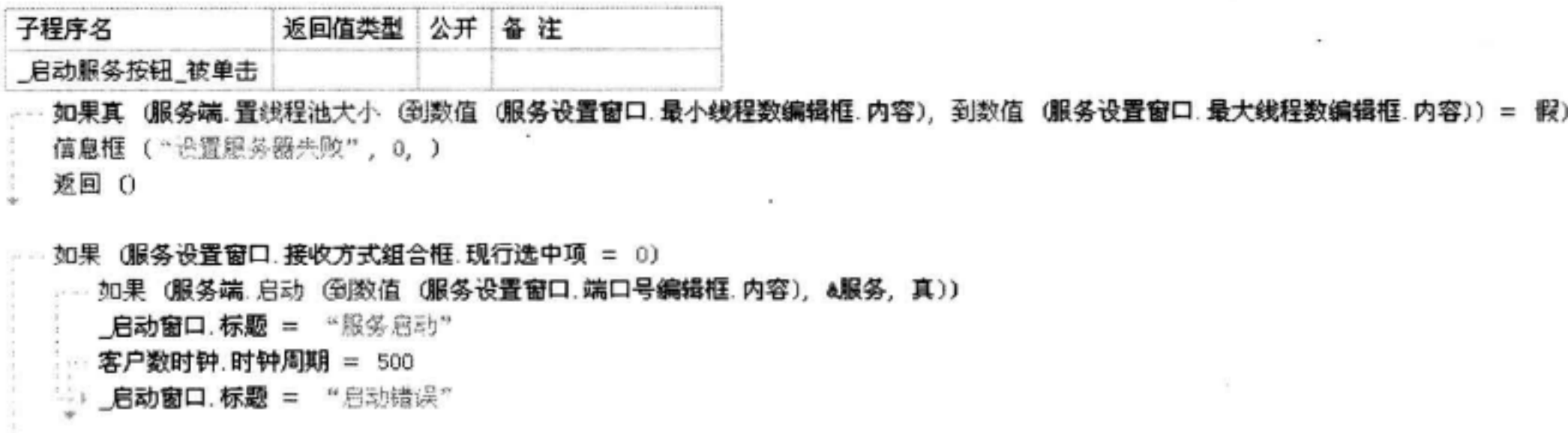


图 13-24 “启动方法”方法

程序执行完后,启动服务按钮被单击,启动远程服务。

2. “停止()”方法

该方法用来停止正在工作的服务程序,停止服务后会释放大量的资源。

【注意】由于要释放大量的系统资源,该方法执行过程中会有一点延时,但一定要等此方法执行完毕后再关闭主程序,如果强行关闭主程序而没有执行完该方法,则会引起大量的内存资源泄露,甚至会出现非法错误。所以,可以在“_启动窗口_可否被关闭”或“_启动窗口_将被销毁”事件子程序中执行该方法。如图 13-25 所示。

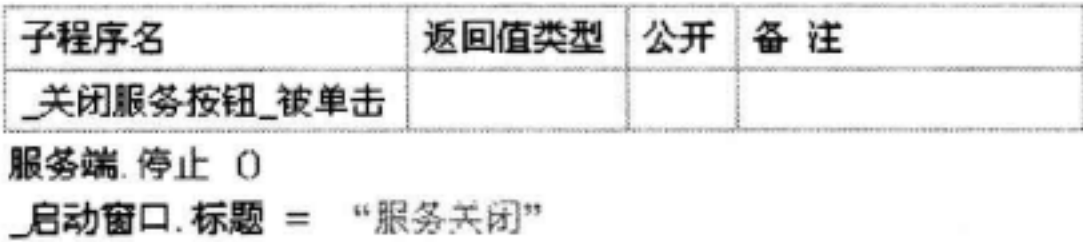


图 13-25 “停止”方法

程序执行完后,单击关闭服务按钮,则服务端停止所有服务,启动窗口标题为“服务关闭”。

3. “置线程池大小()”方法

设置每一个线程池内的线程数量的最大值和最小值,该项设置必须在服务器启动之前设置。服务系统拥有四个线程池:一个接收线程池,一个发送线程池,还有两个处理线程池,该方法是设置每一个线程池内的允许线程数量。由于线程池内的线程数量是可以随环境的变化而增加或减少的,所以该方法也就是设置线程池内的线程数量的变化区间,系统会根据该设置取最优化的值。如果不进行该项设置,系统将采用默认的设置,即最小线程数量为 5,最大线程数量为 20。设置成功返回“真”,失败返回“假”。如图 13-26 所示。

子程序名	返回值类型	公开	备注
启动服务按钮_被单击			
如果真 (服务端.置线程池大小 (到数值 (服务设置窗口.最小线程数编辑框.内容), 到数值 (服务设置窗口.最大线程数编辑框.内容)) = 假) 信息框 ("设置服务器失败", 0,) 返回 0			

图 13-26 “置线程池大小”方法

程序执行完后,服务端设置每一个线程内的线程的最大值和最小值。

4. “取客户句柄()”、“取客户 IP()”方法

“取客户句柄()”方法根据客户的请求信息地址取得客户的句柄,有 2 个参数,第 1 个参数为“请求信息地址”,该地址为“启动()”方法指定的“处理函数”的参数;第 2 个参数要填入一个整数型变量,该变量用来存放取得的客户句柄。

“取客户 IP()”方法根据客户的句柄值取得客户的 IP 地址,有 2 个参数,第 1 个参数是用“取客户句柄()”方法取得的客户句柄;第 2 个参数填入一个文本型变量,用来存放取得的客户 IP。如图 13-27 所示。

变量名	类型	静态	数组	备注
客户IP地址	文本型			
客户句柄	整数型			
如果真 (服务端.取客户句柄 (请求信息地址, 客户句柄)) 如果真 (服务端.取客户IP (客户句柄, 客户IP地址)) 记录编辑框.内容 = 记录编辑框.内容 + 客户IP地址				

图 13-27 “取客户句柄”方法

程序执行完后,在“服务”子程序中,为“启动”方法指定的“处理函数”,当接收到客户发送的数据时,会触发该子程序。接收到消息后则使用“取客户句柄()”方法取得当前发送数据客户的客户句柄,然后使用该句柄取得发送数据客户的 IP 地址,并显示在“记录编辑框”中,然后运行“取文本()”和“取字节集()”子程序。这两个子程序用来取得客户发送来的文本或字节集。

5. “取请求文本()”方法

“取请求文本()”方法,用来取得客户端发送的文本数据,有 3 个参数,第 1 个参数为“启动()”方法指定的“处理函数”的参数;第 2 个参数为“请求代码”,填写一个文本型变量,用来存放服务器所接收的请求信息来源,该代码唯一标识客户端的一次请求;第 3 个参数为“请求信息”,填写一个文本型变量,用来存放取得的客户端发送的文本。如图 13-28 所示。

程序执行完后,在“服务”子程序中,调用“取文本”子程序,即当客户端有数据到达时,则使用“取文本”子程序取得客户端发送的文本信息,子程序中,使用“取请求文本()”方法取的客户端发送的文本,成功后则将取得的文本显示在“记录编辑框”中,并使用“发送文本()”方法,向客户端发送信息。

6. “取请求字节集()”方法

用来取得可户端发送的字节集数据,有 3 个参数,前 2 个参数和“取请求文本()”方法前 2 个参数使用方法相同,第 3 个参数为“请求信息”,填写一个字节集型变量,用来存放取得的字节集数据。如图 13-29 所示。

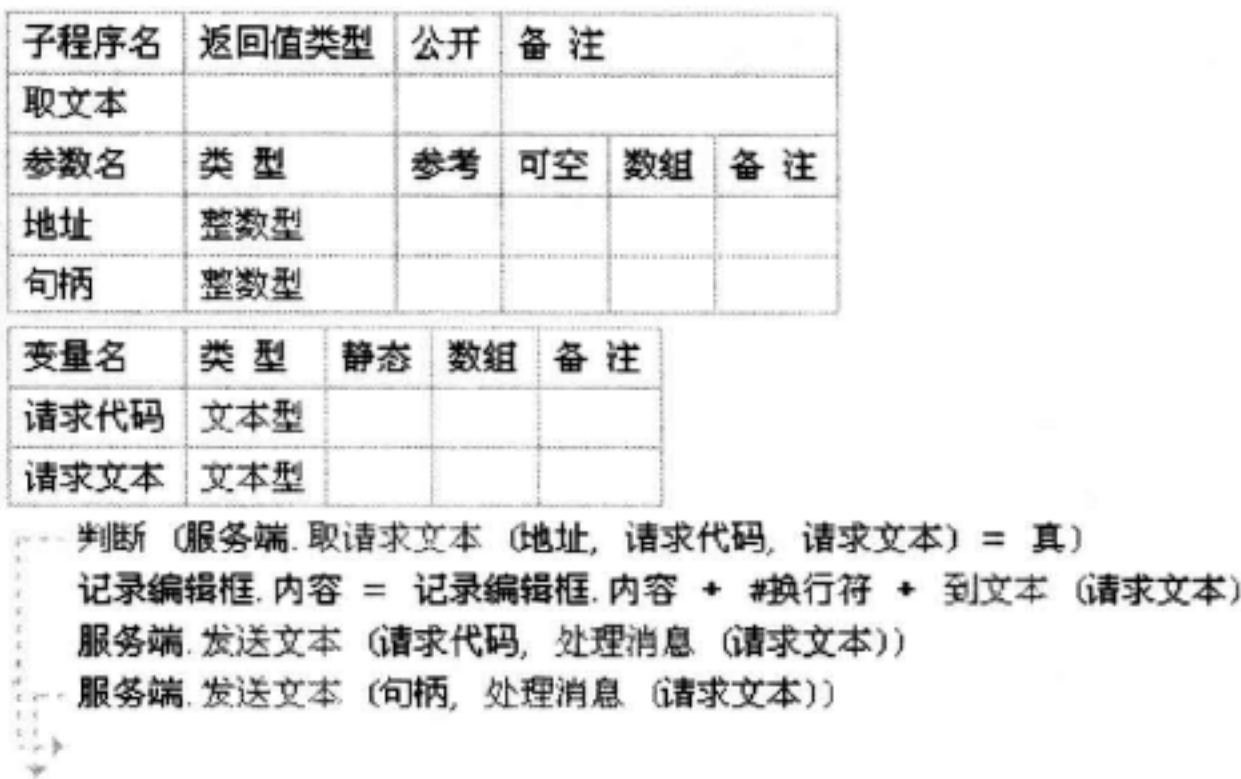


图 13-28 “取请求文本”方法

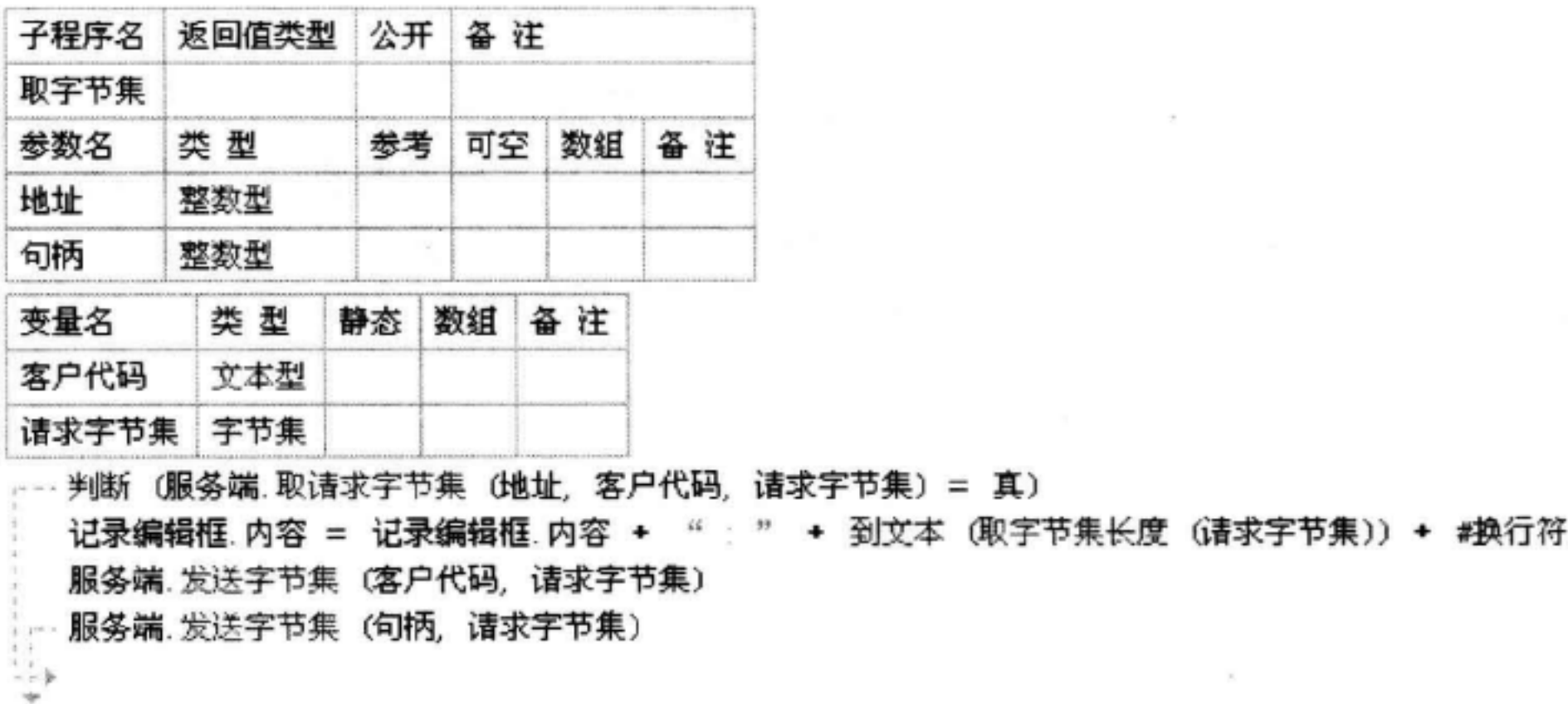


图 13-29 “取请求字节集”方法

程序执行完后,该子程序对客户端发送的文本进行判断,根据客户端发送的不同文本而执行不同的操作,执行成功,则返回执行成功的消息文本,如果客户端发送的不是服务端指定的命令,则返回“非命令文本”。该子程序的返回值将会使用“发送文本()”方法发送给客户端。

7. “发送文本()”方法

“发送文本()”服务程序发送文本数据到客户端。该方法执行成功返回“真”,失败返回“假”。

语法结构是:逻辑型 远程服务.发送文本(目的端,发送信息)

其中,目的端是必需的,通用型。该参数可以接受两种不同的数据,一个是“请求代码”,一个是“客户句柄”,这两种数据都可以作为服务程序发送数据的目的端,当参数为“请求代码”时,发送的目的端就是该请求代码的来源。发送信息也是必需的,文本型。服务器所要发送的文本形式的信息。如图 13-30 所示。

程序执行完后,如果服务端取请求文本成功,服务端分别发送文本到请求代码来源处和句柄客户。

8. “发送字节集()”方法

服务程序发送字节集数据到客户端。该方法执行成功返回“真”,失败返回“假”。

子程序名	返回值类型	公开	备 注		
取文本					
参数名	类 型	参 考	可 空	数 组	备 注
地址	整数型				
句柄	整数型				

变量名	类 型	静 态	数 组	备 注
请求代码	文本型			
请求文本	文本型			

判断 (服务端.取请求文本 (地址, 请求代码, 请求文本) = 真)

记录编辑框.内容 = 记录编辑框.内容 + #换行符 + 到文本 (请求文本)

服务端.发送文本 (请求代码, 处理消息 (请求文本))

服务端.发送文本 (句柄, 处理消息 (请求文本))

图 13-30 “发送文本”方法

语法:逻辑型远程服务.发送字节集(目的端,发送信息)

其中,目的端是必需的,通用型。该参数可以接受两种不同的数据,一个是“请求代码”,一个是“客户句柄”,这两种数据都可以作为服务程序发送数据的目的端,当参数为“请求代码”时,发送的目的端就是该请求代码的来源。如图 13-31 所示。

子程序名	返回值类型	公开	备 注		
取字节集					
参数名	类 型	参 考	可 空	数 组	备 注
地址	整数型				
句柄	整数型				

变量名	类 型	静 态	数 组	备 注
客户代码	文本型			
请求字节集	字节集			

判断 (服务端.取请求字节集 (地址, 客户代码, 请求字节集) = 真)

记录编辑框.内容 = 记录编辑框.内容 + “: ” + 到文本 (取字节集长度 (请求字节集)) + #换行符

服务端.发送字节集 (客户代码, 请求字节集)

服务端.发送字节集 (句柄, 请求字节集)

图 13-31 “发送字节集”方法

程序执行完后,如果服务端取请求字节集成功,服务端就发送字节集到客户代码来源和客户句柄处。

9. “取客户数()”方法

“取客户数()”方法取得当前连接在服务器端的客户端数量。如图 13-32 所示。

程序执行完后,服务端取当前连接在服务端的客户端数量,如果客户数大于等于 0,启动窗口标题显示为:标题变量+“-----”当前客户数。

10. “取客户数组()”方法取得当前连接到服务器的所有客户端的句柄,在参数中提供一个整数型数组,来存放取得的客户句柄数组。

11. “断开连接()”方法

该方法用来断开指定客户端的连接,在参数中提供欲断开客户端的客户句柄,客户句柄通

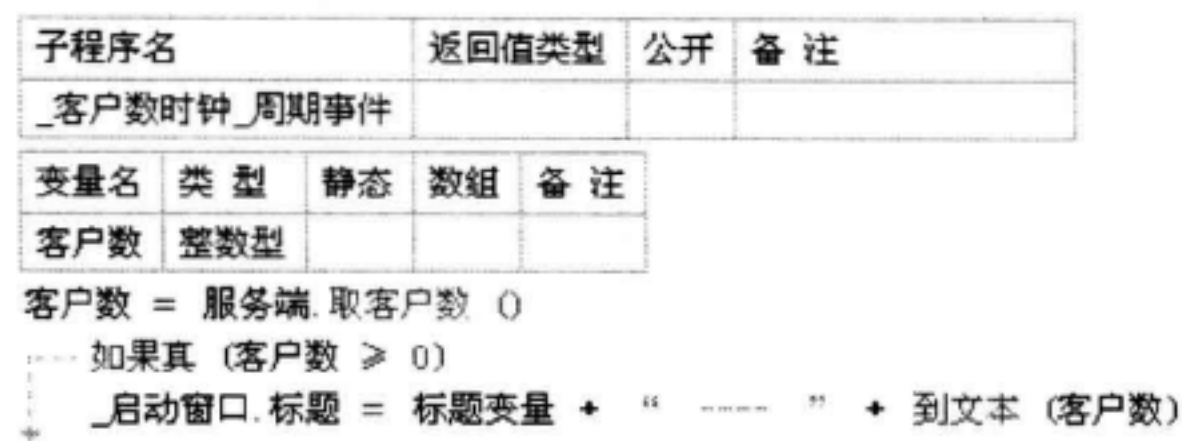


图 13-32 “取客户数”方法

过“取客户句柄()”方法取得。如图 13-33 所示。

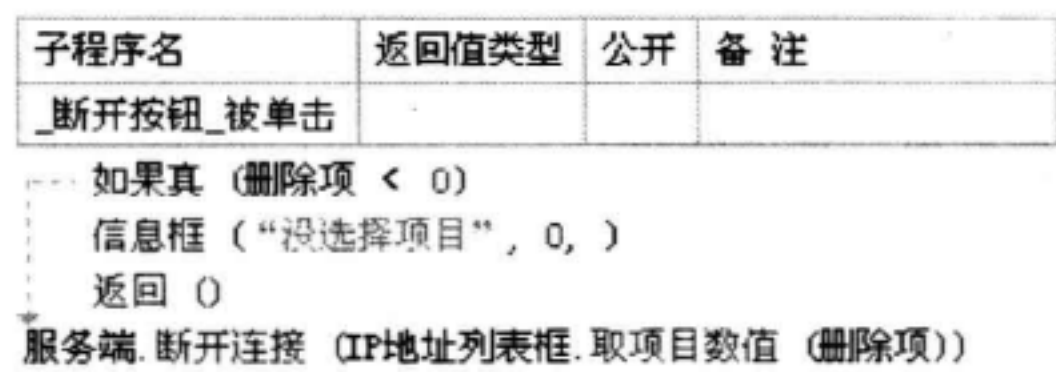


图 13-33 “断开连接”方法

程序执行完后,单击“断开”按钮,断开指定客户端的连接。

12. “取消息类型()”方法

取得触发服务端处理函数的消息类型。有客户连接到服务端返回 0,有客户主动或意外断开与服务端的连接返回 1,客户端发送字节流到服务器返回 2,出错返回 -1。

语法:整数型 远程服务.取消息类型(消息地址)

其中,消息地址是必需的,整数型。触发服务器端处理函数的消息地址。

13.5.3 请求客户端的方法

1. “连接()”方法

“连接()”方法将客户端连接到指定地址和端口号的服务器。

调用格式:〈逻辑型〉请求客户端.连接(端口号,服务器地址,请求方式,[接收函数])

其中,端口号,整数型,本参数作为欲连接的服务器的端口号,该值的取值范围在 0 ~ 65535 之间。

【注意】客户端要连接与服务端相同的端口号。

服务器地址,文本型,本参数是欲连接的服务器的 IP 地址,也可以是服务器的域名或者是内网的计算机名。

请求方式,逻辑型,本参数作为客户端的请求方式,该值为真代表客户端采用同步的请求方式,这时忽略该方法的最后一个参数,该值为“假”代表客户端采用异步的请求方式。同步方式不能发送异步请求,同样异步方式不能发送同步请求。

接收函数,子程序指针,客户端向服务器发送异步请求后,当客户端接收到服务器返回的结果,该函数就会自动被系统调用执行。该函数有一个整型的参数,该参数代表服务器返回的结果地址,客户端可以通过该地址得到返回的结果内容。如图 13-34 所示。

客户端程序中,“发送方式组合框”提供“同步连接”和“异步连接”选项,用于选择与服务端的连接方式,“连接按钮”被单击后首先根据“发送方式组合框”的当前选项,选择与服务器

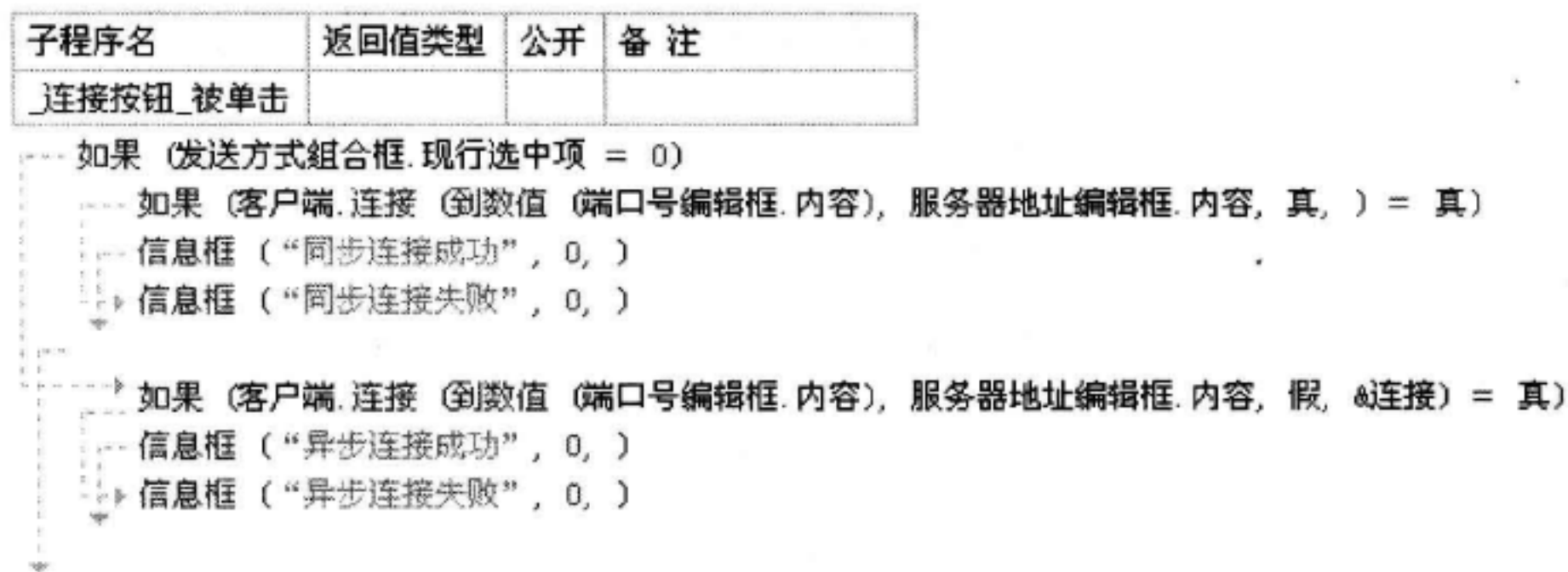


图 13-34 “连接”方法

端的连接方式,然后用信息框提示连接结果。

【注意】当客户端选择同步连接服务器端时,应使用“同步发送文本()”或“同步发送字节集()”方法向服务端发送数据,这两种方法直到接收到服务端发送的数据后才返回;如果客户端选择异步连接服务端,则应使用“异步发送文本()”或“异步发送字节集()”方法向服务端发送数据,这两个方法不等待服务端发送的数据而直接返回。

2. “断开()”方法,将客户端与服务端的连接断开。如图 13-35 所示。

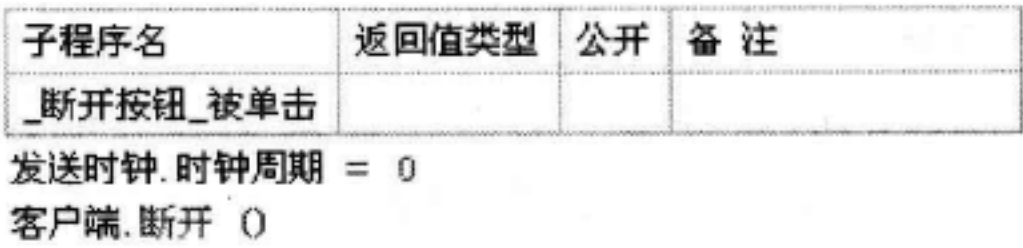


图 13-35 “断开”方法

程序执行完后,单击“断开”按钮,客户端与服务端自行断开。

3. “同步发送文本()”方法

“同步发送文本()”方法在客户端发送同步请求文本到服务器,当收到服务器返回的结果后,该方法才返回,否则为阻塞状态直到超时。

语法: 整数型 请求客户端.同步发送文本(请求文本,结果文本,超时)

其中,“请求文本”是文本型,指客户端发送的请求文本的内容。结果文本也是文本型,参数数据只能提供变量,指的是服务器返回的结果内容。超时是整数型,同步机制下“发送请求”方法的最大允许等待时间,单位为毫秒。该值必须大于等于0。

4. “同步发送字节集()”

方法在客户端发送同步请求字节集到服务器,当收到服务器返回的结果后,该方法才返回,否则为阻塞状态直到超时。

语法: 整数型 请求客户端.同步发送字节集(请求字节集,结果字节集,超时)

其中,请求字节集指客户端发送的请求字节集的内容。是字节集型。结果字节集,参数数据只能提供变量。服务器返回的结果内容。超时是整数型。同步机制下“发送请求”方法的最大允许等待时间。单位为毫秒。该值必须大于等于0。

这两个方法的使用方法类似,以“同步发送文本()”方法为例进行说明,如图 13-36 所示。

程序执行完后,发送文本按钮被单击,程序中首先判断当前连接为同步连接后,则使用

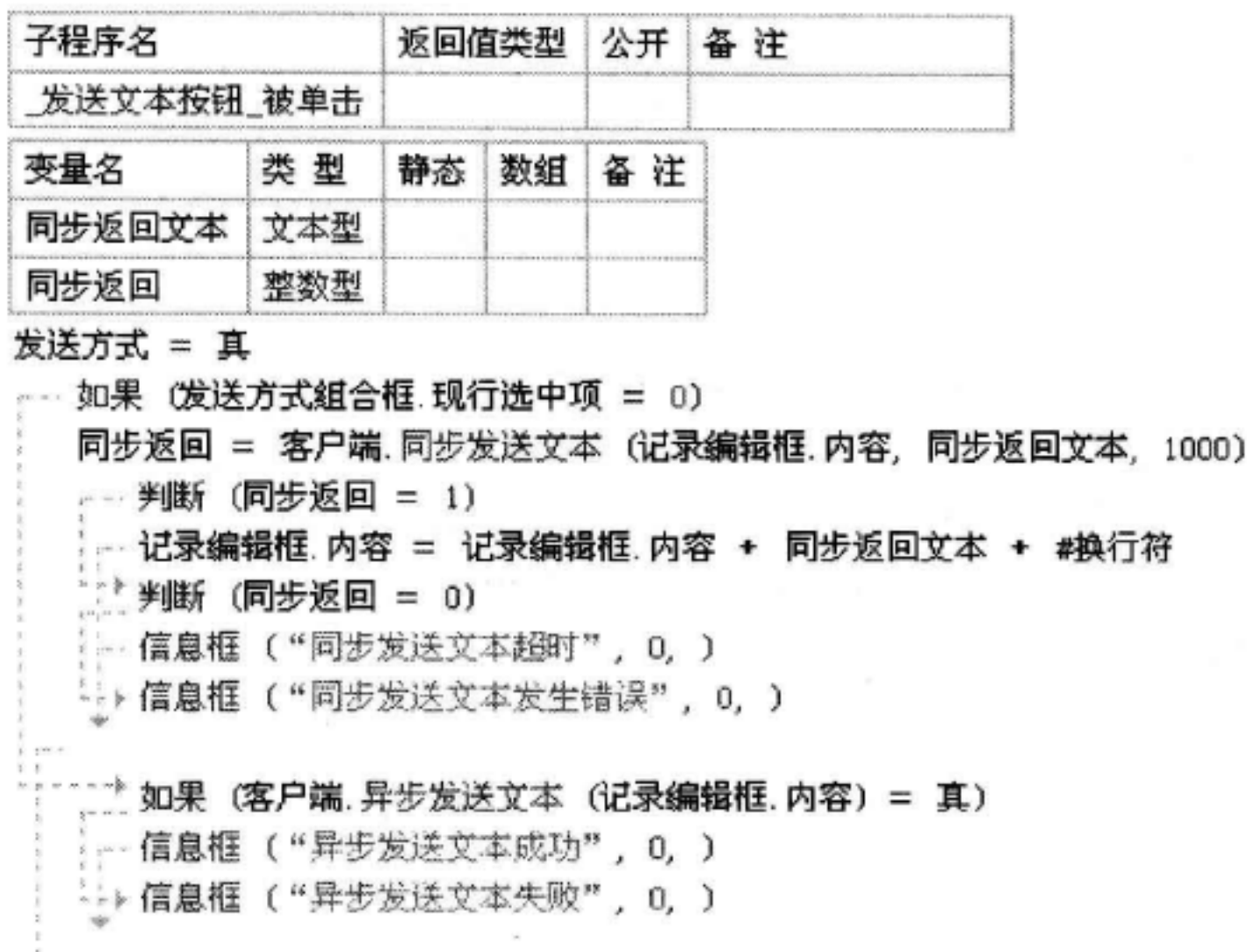


图 13-36 “同步发送文本”方法

“同步发送文本()”方法,向数据库发送文本,这里要注意“同步发送文本()”方法的返回值是整数型的,1 代表发送成功,0 代表发送超时,其他数字代表发生错误,所以程序中的,当返回值是 1 的时候,则将服务端同步返回的文本显示在“记录编辑框”中。

【注意】当使用同步发送方式时,客户端将不会使用“连接()”方法指定的子程序来接收服务端返回的数据。

5. “异步发送文本()”方法

客户端发送异步请求文本到服务器,发送完毕后,该方法立刻返回。该方法执行成功返回“真”,执行失败返回“假”。语法:

逻辑型 请求客户端. 异步发送文本(请求信息)

其中,请求信息是文本型,指客户端发送的请求信息的内容。如图 13-37 所示。

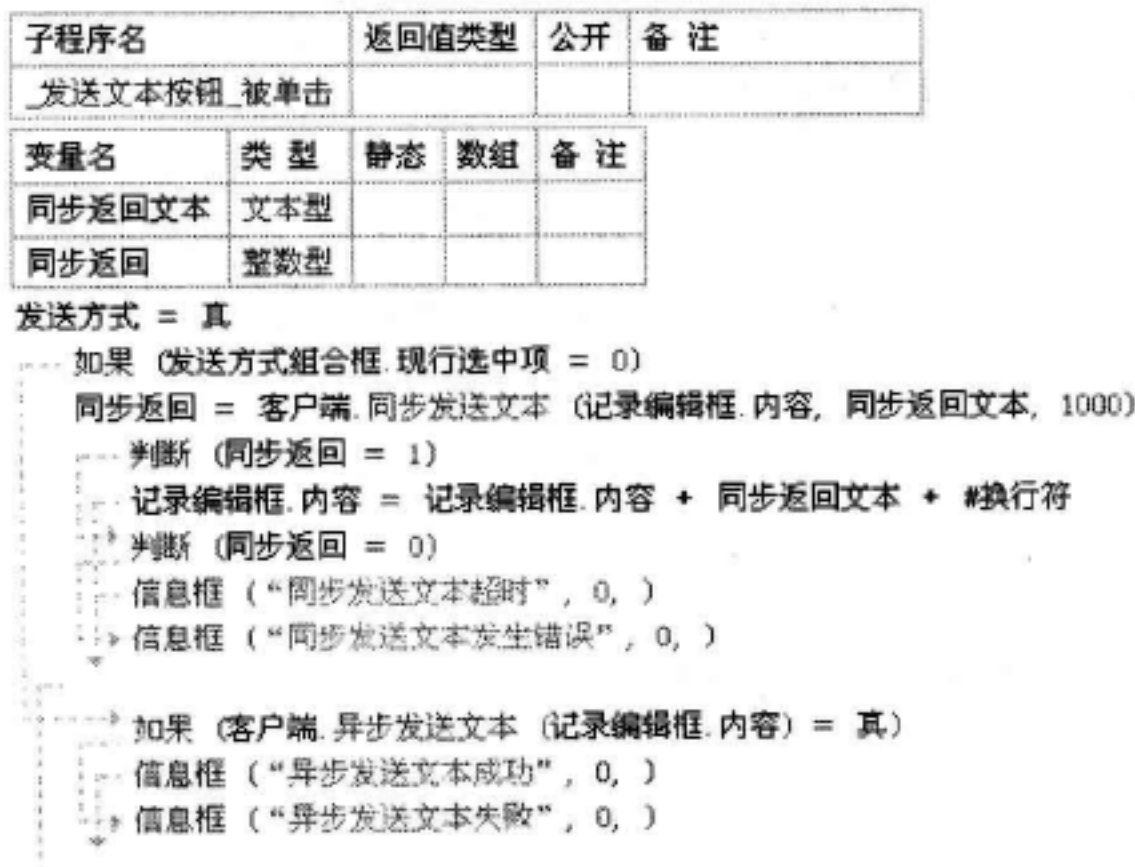


图 13-37 “异步发送文本”方法

程序执行完后,发送文本按钮被单击,程序中首先判断当前连接为异步连接后,则使用“异步发送文本()”方法,向数据库发送文本。

6. “异步发送字节集”

因其与“异步发送文本”类似,这里不再详述。

7. “取返回文本()”方法

在异步的请求方式下,取得服务器返回的结果。如果正确取得服务器的返回结果,该方法返回“真”,否则,该方法返回“假”。

语法:逻辑型 请求客户端.取返回文本(结果信息地址,结果信息)

其中,结果信息地址是整数型。服务器发送回来的结果信息的地址。“取返回文本”方法必须在用户定义的“接收函数”中调用,所以该值就是“接收函数”的参数值。

结果信息,文本型,参数数据只能提供变量。客户端得到的服务器返回的结果信息,该参数作为方法“取返回文本”的返回值。

8. “取返回字节集()”方法

“取返回文本()”方法与“取返回字节集()”方法类似,在这里也不再详述,关于两个方法的使用,如图 13-38 所示。

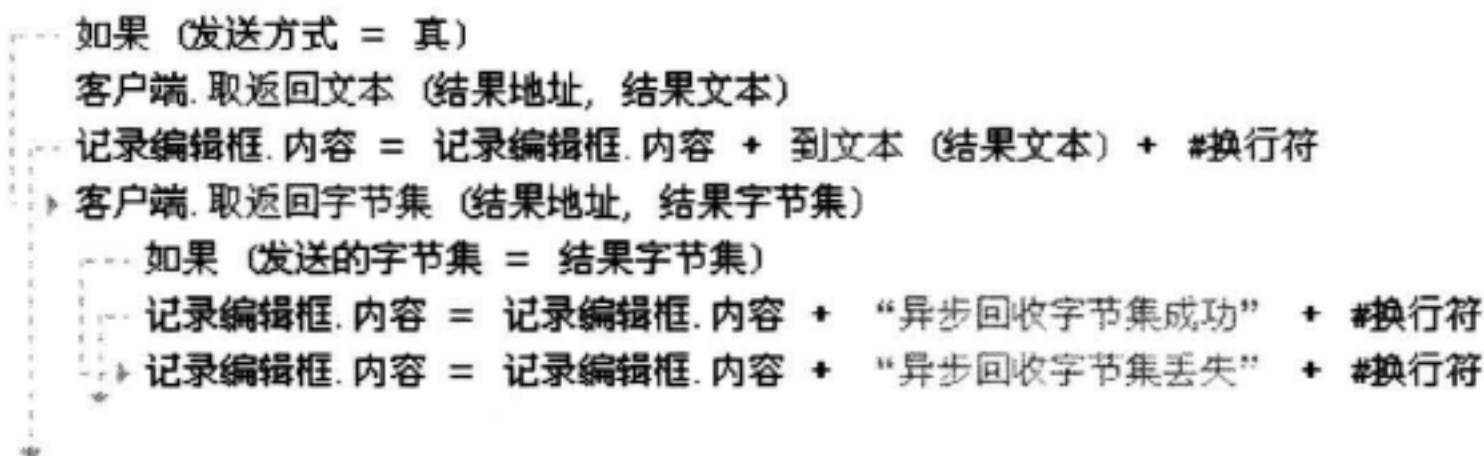


图 13-38 “异步发送文本”方法

程序执行完后,如果是异步请求方式,客户端取得服务器的返回结果。

本节中引用的代码参见例程 13-5。

13.6 网络传送支持库

13.6.1 网络传送支持库概述

网络传送支持库实现对多种协议断点续传下载与 FTP 上传的支持,并提供了丰富的设置方式。当前版本支持 HTTP,FTP,MMS 多线程下载及断点续传,以及对 RTSP 协议的单线程下载及断点续传。

网络传送支持库也是以面向对象的编程思路编写的,所以用户使用的时候也要创建支持库中提供的各种对象和使用对象提供的方法。

13.6.2 网络传送支持库中的对象

网络传送支持库中提供了 3 个对象类型,分别为:“下载对象”、“FTP 上传对象”、“FTP 辅助对象”。每个对象都有提供的方法,并负责不同的上传和下载任务。

“下载对象”提供的方法用来新建 HTTP 和 FTP 的下载任务,并可以在下载过程中返回下载信息和日志;“FTP 上传对象”负责对 FTP 服务器端的文件上传;“FTP 辅助对象”提供的方法用来连接 FTP 服务器和对 FTP 服务器端的目录进行操作等。

在本支持库中,还包括了一条独立于其他对象之外的命令,该命令为全局命令“网络通讯设置()”,在使用本支持库前,必须使用该命令来初始化。其语法结构是:

逻辑型 网络通讯设置(写入文件大小,重连时间,重连次数)

其中,写入文件大小是整数型,初始值为“20480”。本参数指定每次写入文件的指定大小,默认为 20480。重连时间也是整数型,初始值为“0”。出错时连接服务器间隔时间,默认为 0。重连次数,也是整数型,初始值为“0”,出错时重复连接服务器次数,默认为 0。如图 13 - 39 所示。

子程序名		返回值类型	公开	备注
__启动窗口_创建完毕				
变量名	类型	静态	数组	备注
变量	文本型			

网络通讯设置 (到数值 (写入文件大小,内容), 0, 0)

图 13 - 39 “网络通讯设置”方法

程序执行完后,网络通讯设置 FTP 的上传下载和速度的指令,其中重连时间和重连次数都是 0。

13.6.3 “下载对象”的方法

1. “增加新任务()”方法

功能:该方法用来为下载对象新增加一个下载任务。

语法:〈逻辑型〉下载对象.增加新任务(配置信息)

参数:“配置信息”为“任务参数”数据类型,任务参数类型的数据包括了多个成员,用于设置下载的相关信息,表示方法为“任务参数.成员名称”,下载时必须设置的成员有:

“类型”,指定任务类型。可以为枚举常量“传送常量”中的以下成员:下载、上传、覆盖、续传、改名、跳过、被动选择,表示方法为“#传送常量.下载”

“文件地址”,定欲读取文件在互联网上的地址,即(URL)。

“本地文件路径”,本成员指定被下载或待上传文件的本地保存位置,当作为下载文件保存路径使用时。如果没有给定文件名,那必须是一个以“\”结尾的文件路径。系统会根据“文件地址”指定的文件名作为本地保存的文件名,系统会处理重定向情况。当收到“下载或上传开始”消息后可以通过“取本地文件路径”获得包含文件名的文件路径。

“用户名”和“密码”只用做 FTP 登陆时使用。

“日志回调函数”,该成员很重要,必须赋值一个子程序指针型的数据,即按照下面解释的子程序结构新建一个子程序,并将该子程序的指针赋值给“任务参数.日志回调函数”。

“日志回调函数”有五个参数,第一个参数(整数型),线程信息,可以通过“取线程信息”获得每个上传或下载对象的线程信息,与本参数比较从而确定发送本消息的对象。第二个参数(文本型),消息产生的时间。第三个参数(文本型),消息的正文。第四个参数(整数型),消息类型,参见“传送常量”。第五个参数(整数型),对象信息,可以通过“取对象信息”获得下载或上传对象的信息与本参数比较,从而确定本消息属于哪个对象。

“交互回调函数”,该成员非常重要,必须指定必须赋值一个子程序指针型的数据,和“日志回调函数”的赋值方法相同,但子程序的参数类型和日志回调函数不同。

“交互回调函数”有五个参数,第一个参数(整数型)消息类型,参见“传送常量。”第二个

参数(整数型),线程信息,可以通过“取线程信息”获得每个上传或下载对象的线程信息,与本参数比较从而确定发送本消息的对象第三个参数(整数型)根据消息类型的不同而不同,参见具体的消息第四个参数(整数型)根据消息类型的不同而不同,参见具体的消息。第五个参数(整数型),对象信息,可以通过“取对象信息”获得下载或上传对象的信息与本参数比较,从而确定本消息属于哪个对象。如图 13-40 所示。

子程序名	返回值类型	公开	备注
增加新任务_被单击			
下载参数.类型 = #传送常量.下载			
下载参数.文件地址 = 地址栏.内容			
下载参数.下载速度 = 到数值(速度框.内容)			
下载参数.线程等待超时 = 1000			
下载参数.默认线程数 = 到数值(线程数.内容)			
线程数.内容 = "0"			
下载参数.本地文件路径 = 本地路径.内容			
下载参数.连接超时 = 5000			
下载参数.发送超时 = 5001			
下载参数.接收超时 = 10000			
下载参数.接收大小 = 到数值(接收大小框.内容)			
下载参数.交互回调函数 = &交互回调函数			
下载参数.日志回调函数 = &日志回调函数			
下载参数.用户名 = 用户名.内容			
下载参数.用户密码 = 密码.内容			
下载任务.增加新任务(下载参数)			

图 13-40 “增加新任务”方法

程序执行完后,单击增加新任务按钮,为系统增加一个新任务,并配置相应参数信息。

2. “停止指定任务()”

功能:停止一个正在进行的下载任务,并保存任务信息,任务信息用来继续下载时使用。

语法:〈逻辑型〉下载对象.停止指定任务(任务数据,执行方式)

参数:“任务数据”为字节集型,必须填写一个字节集变量,用于保存任务信息,如果在程序结束后希望再次打开程序时,仍可以继续已停止的下载,则可以将任务信息保存在数据库中;“执行方式”为整数型,1 表示阻塞方式,即直到所有停止工作全部做完后返回,如果在指定的时间段内还没有全部结束则强制关闭线程并返回,这个时间段为“任务参数”中的“线程等待超时”,2 表示非阻塞方式即结束操作由系统在后台处理,直接返回,默认为 2。如图 13-41 所示。

子程序名	返回值类型	公开	备注
停止下载任务_被单击			
变量名	类型	静态	数组
已下载长度	整数型		
如果真(下载任务.停止指定任务(下载信息,到数值(2)) = 假)			
信息框(“停止错误”,0,“错误”)			

图 13-41 “停止任务”方法

程序执行完后,单击停止下载任务按钮,即停止正在进行的下载任务,并保存任务信息,若停止下载任务为假,则信息框显示“停止错误”。

3. “继续下载任务()”方法

功能:继续一个下载文件的任务,当前支持 HTTP、FTP 协议。成功返回“真”,失败返回“假”。

【注意】继续下载时默认线程数不是根据任务参数中的默认线程数进行设置的,而是根据任务信息中的没有完成的断点信息的数量进行设置的。断点的数量则是根据任务自本次运行以前开启的线程的总数减去已经完成下载任务的线程数获得的。

语法:〈逻辑型〉下载对象.继续下载任务(配置信息,任务数据)

参数:“配置信息”为“任务参数”数据类型,和“增加新任务()”方法中的任务参数相同的使用方法;“任务数据”为字节集型,是“停止指定任务()”方法保留的任务数据。如图 13-42 所示。

子程序名	返回值类型	公开	备注
_继续下载任务_被单击			

下载任务,继续下载任务 (下载参数, 下载信息)

图 13-42 “继续下载任务”方法

程序执行完后,单击继续下载任务,则继续下载先前停止下载的任务。

4. “增加线程()”方法方法

增加一个下载线程,如果最大块的长度小于接收大小的 2 倍时将不能开启新线程注意:下载线程原则上可以无限增加,但是请您根据您的实际情况谨慎使用。不要过多或过于频繁的增加线程,否则不但不能提高效率,相反会影响系统的稳定。

语法:逻辑型下载对象.增加线程(),如图 13-43 所示。

程序运行后,单击增加线程按钮被单击,则增加一个线程。

5. “减少线程()”方法

减少一个下载线程,当只有一个下载线程工作时,将不能结束这个线程。

语法:逻辑型下载对象.减少线程()

“减少线程()”方法也没有参数,每次运行都增加或减少 1 个线程。如图 13-44 所示。

子程序名	返回值类型	公开	备注
_增加线程_被单击			

如果 (下载任务.增加线程 () = 真)
命令返回框.内容 = “增加线程 结果 :真”
命令返回框.内容 = “增加线程 结果 :假”

图 13-43 “增加线程”方法

子程序名	返回值类型	公开	备注
_减少线程_被单击			

如果 (下载任务.减少线程 () = 真)
命令返回框.内容 = “减少线程 结果 :真”
命令返回框.内容 = “减少线程 结果 :假”

图 13-44 “减少线程”方法

程序运行后,单击减少线程按钮则减少一个下载线程。

6. “取下载速度()”、“取已下载长()”、“取重试次数()”方法

下载速度、已下载文件的大小和重试次数,是网络下载经常查看的几个信息,这 3 种方法就是用来取得这些信息。

这 3 种方法的使用都很简单,没有参数,返回值便是取得的结果,返回值都是整数型,但是单位不同。“取下载速度()”和“取已下载长()”方法取得的结果是以字节/秒为单位。如图 13-45 所示。

程序执行完后,刷新显示当前所有下载任务的下载速度、已下载长及重试次数。直接将取得的结果转换成文本依次显示在对应的编辑框中。

7. “取任务数据()”方法

“取任务数据()”方法,获得任务数据。该方法有一个字节集参数,提供一个字节集变量

子程序名	返回值类型	公开	备注
__时钟1_周期事件			
_启动窗口.标题 = “总下载速度:” + 到文本(取总下载速度 0)			
下载速度.内容 = “速度:” + 到文本(下载任务.取下载速度 0)			
已经下载.内容 = “已经下载:” + 到文本(下载任务.取已下载长 0)			
重试次数.内容 = “重试次数:” + 到文本(下载任务.取重试次数 0)			

图 13-45 “取下载速度”方法

用来保存取得的任务数据。任务数据是用来提供给“继续下载任务()”方法继续下载时使用的,和“停止指定任务()”方法运行时保存的任务数据是相同的。主要是提供给用户随时备份任务数据时使用,这样可以在程序异常退出后,仍保证未下载完的任务可以继续。可以使用时钟定期执行该命令来保存任务数据。例如:

命令返回框.内容 = “取任务信息” + 到文本(下载任务.取任务数据(下载信息))

8. “取对象信息()”方法

“取对象信息()”方法取得的对象信息是一个整数值,主要使用在“交互回调函数”或“日志回调函数”中,将取得的对象信息与这两个函数的第 5 个参数进行比较,从而可以确定是哪个对象返回的信息,这是因为多个下载对象共用一个“交互回调函数”和“日志回调函数”。例如:

回调消息框.加入文本(“对象信息:第” + 到文本(下载任务.取对象信息()) + “个下载对象”)

9. “取线程信息()”方法

“取线程信息()”方法可以取得当前下载对象所有的线程信息。该方法有一个整数型参数,提供一个数组型变量,用来保存取得的线程信息组。取得数组的成员数便是当前下载任务正在下载的线程数。例如:

回调消息框.加入文本(“线程信息:第” + 到文本(下载任务.取线程信息(线程组)) + “个下载线程”)

10. “取任务数据项()”方法

前面介绍的“取任务数据()”方法和“停止指定任务()”方法都可以返回一个字节集型的任务数据,该任务数据中其实包含:待下载文件总长度、已下载总长度、断点个数等各种信息,使用“取任务数据项”方法便可以取得任务数据中这些数据项的值。

语法:〈整数型〉下载对象.取任务数据项(任务数据,数据类型,类型参数)

参数:“任务数据”字节集型,填写欲取得数据项目值的任务数据;“数据类型”整数型,需要获得的任务信息的类型,如果“类型参数”作为指定具体断点序号使用,那么它从 1 开始不大于断点个数,1 待下载文件总长度,2 已下载总长度,3 断点个数,4 指定断点开始下载位置,类型参数为序号,5 指定断点待下载长度,类型参数为序号,6 指定断点已下载长度,类型参数为序号;“类型参数”整数型,初始值为“0”。根据“数据类型”参数的不同而不同。当需要查询的信息类型没有指定本参数的意义时,本参数无意义。

例如:

已下载长度 = 下载任务.取任务数据项(下载信息,2,0)

11. “取本地文件路径()”方法

“取本地文件路径()”方法可以将当前下载任务的本地文件保存路径取出并返回。

13.6.4 “FTP 辅助对象”的方法

“FTP 辅助对象”提供了连接 FTP 服务器和对 FTP 服务器端的目录及文件操作的相关方法。

1. “连接 FTP 服务器()”方法

“连接 FTP 服务器()”方法用来连接指定的 FTP 服务器。

语法:〈逻辑型〉FTP 辅助对象. 连接 FTP 服务器(FTP 服务器地址,用户名,密码,端口,[日志回调函数],[超时])

参数:“FTP 服务器地址”文本型,填写欲连接的 FTP 服务器的域名或 IP 地址,参数格式:ftp://域名或 ip/, 例如:ftp://www. eyuyan. com/或 ftp://220. 194. 59. 10/;“用户名”文本型,填写登陆的用户名;“密码”文本型,填写登陆密码;“端口”整数型,为登陆端口,多数 FTP 服务器都为标准端口 21,所以该参数一般保持默认值即可;“日志回调函数”子程序指针型,该回调函数的定义方法和“下载对象”中介绍的相同;“超时”整数型,该参数为登陆过程中数据交互的超时时间,单位为毫秒,如果网络状况不好可以将该值设置的稍大些。如图 13 -46 所示。

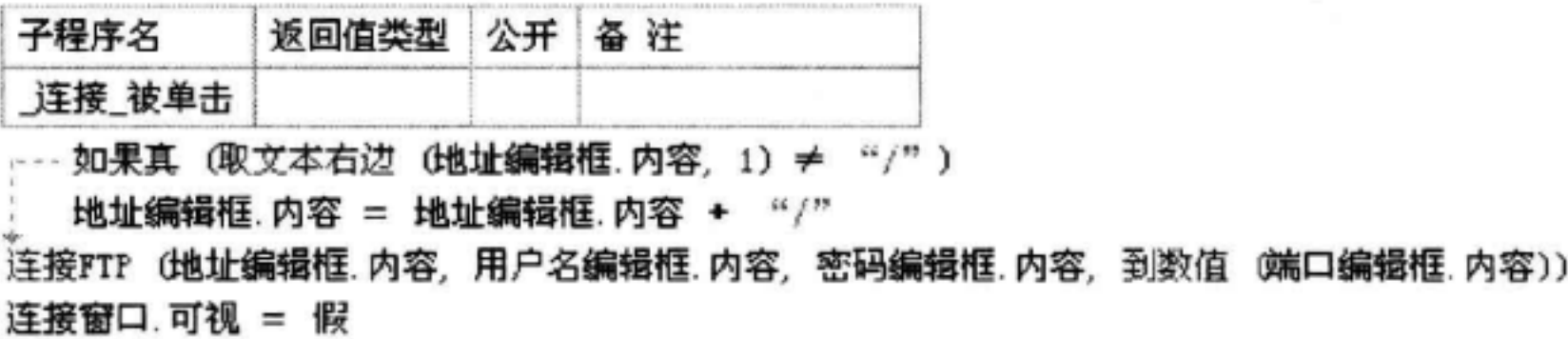


图 13 -46 “连接 FTP”方法

程序执行后,点击单击按钮后,连接指定的 FTP 服务器。

2. “删除文件()”、方法

“删除文件()”方法用来删除 FTP 服务器端的文件。

语法:〈逻辑型〉FTP 辅助对象. 删除文件([欲删除的 FTP 文件])

参数:“欲删除的 FTP 文件”填写准备删除的 FTP 文件,如果不填写服务器端的路径,则删除当前目录的指定文件,如果是填写带有路径的文件名,例如:“模板\临时模板. htm”,则表示从当前目录算起的目录,即当前目录下的“模板”目录中的“临时模板. htm”文件。

例如:当前目录为 FTP 服务端根目录,删除“工具”目录中的“index. htm”文件,代码如下:

FTP 辅助对象. 置现行行目录(“工具/index. htm”)

或者可以使用代码:

FTP 辅助对象. 置现行目录(“工具/”)

FTP 辅助对象. 删除文件(“index. htm”)

上面代码是先用“置现行目录()”方法改变当前目录,然后删除当前目录的文件,这样使用时要注意保存当前目录,以免下次删除文件时出错。

3. “创建目录()”、“删除目录()”、“置现行目录()”、“取现行目录()”方法

这四个方法,用来对 FTP 服务器端的目录进行操作。

“创建目录()”方法,用来在服务器新建目录。

语法:逻辑型 FTP 辅助对象. 创建目录 ([FTP 目录路径])

参数:“FTP 目录路径”文本型,填写要创建的目录名称,这里要注意,只能在当前目录创建目录,不能创建多级目录。

例如:创建一个名为“新建文件夹”的文件,代码如下:

FTP 辅助对象. 创建目录(“新建文件夹”)

4. “删除目录()”方法

该方法用来删除 FTP 服务器中的指定目录及其中所有的文件和文件夹。

语法:逻辑型 FTP 辅助对象. 删除目录([FTP 目录路径])

参数:“FTP 目录路径”文本型,填写要删除的目录名称,这里要注意,删除的目录名也从当前目录开始算起,例如:当前目录为服务器端根目录,要删除根目录中“工具”目录下的“应用软件”目录,代码如下:

FTP 辅助对象. 删除目录(“工具\应用软件\”)

如果当前,目录为“工具”,则代码改为:

FTP 辅助对象. 删除目录(“应用软件\”)

5. “置现行目录()”方法

该方法用来设置 FTP 服务器上的当前目录。

语法:逻辑型 FTP 辅助对象. 置现行目录([FTP 目录路径])

参数:“FTP 目录路径”文本型,填写设置为当前的目录的路径名。和前几个命令相同,该参数的路径名也是从当前目录开始算起。

例如:设置当前目录为“工具”目录下的“应用软件”目录,代码如下:

FTP 辅助对象. 置现行目录(“工具\应用软件\”)

6. “取现行目录()”方法

该方法取得当前目录的目录名,并返回。返回的文本便是取得的结果。

7. “取目录列表()”方法

该方法返回 FTP 服务器上指定目录内的所有匹配文件和子目录信息。成功返回被找到的文件和子目录的数目,失败返回 -1 失败的原因有可能是 FTP 服务器的目录结构暂时不被支持。

【注意】该方法暂时支持的目录结构有限,所以,当运行完该方法后,返回的日志文本为“暂时不支持的列表结构”时,表示当前 FTP 服务器的目录结构列表方式暂时不支持。

语法:〈整数型〉FTP 辅助对象. 取目录列表([FTP 目录路径],[存放文件名的数组变量],[存放文件属性的数组变量],[存放文件尺寸的数组变量],[存放文件时间的数组变量])

参数:“FTP 目录路径”文本型,填写要取得目录结构的路径名,该参数的路径名也是从当前目录开始算起,如果省略则默认为取得当前目录的目录结构;“存放文件名的数组变量”文本型,参数数据只能提供变量及变量数组。提供参数数据时只能提供变量数组。在命令执行完毕后,本变量数组内被顺序填入所找到的匹配文件和子目录名称。变量数组内原有数据被全部销毁,变量数组的维数被自动调整为所找到的文件数目。

“存放文件属性的数组变量”整数型,提供一个数组变量保存取得的文件属性数组;“存放文件尺寸的数组变量”长整数型,提供一个长整数型变量保存返回的文件尺寸数组;“存放文件时间的数组变量”日期时间型,参数数据只能提供变量及变量数组。提供参数数据时只能提供变量数组。在命令执行完毕后,本变量数组内被顺序填入所找到的匹配文件的最后修改

时间,并与文件名数组成员一一对应。变量数组内原有数据被全部销毁,变量数组的维数被自动调整为所找到的文件数目。如图 13-47 所示。

```
--- 如果真 (取文本右边 (远程路径编辑框.内容, 1) ≠ "/" )
    远程路径编辑框.内容 = 远程路径编辑框.内容 + "/"
    远程文件超级列表框.全部删除 ()
    远程文件超级列表框.插入表项 (-1, "上级目录", 0, , , )
    文件数 = FTP辅助对象.取目录列表 ( 文件名, 文件属性, 文件尺寸, 文件时间)
```

图 13-47 “取目录列表”方法

13.6.5 “FTP 上传对象”的方法

1. “上传文件()”、“停止上传()”方法

“上传文件()”方法,用于开始一个上传任务。

语法:〈逻辑型〉FTP 上传对象.上传文件 (配置信息)

参数:“配置信息”任务参数型,提供一个“任务参数”型数据。该参数和“下载对象”中的“增加新任务()”方法的参数使用方法相同。

“停止上传()”方法,将正在上传中的任务停止。该方法没有参数,运行即可,成功返回“真”,失败返回“假”。

例如:使用“上传文件()”方法上传超级列表框选中的选项,使用“停止上传()”方法,停止正在上传的文件。如图 13-48 所示。

子程序名	返回值类型	公开	备注
上传按钮_被单击			

变量名	类型	静态	数组	备注
配置信息	任务参数			
欲上传文件名	文本型			文件或目录路径

```
--- 如果真 (本地文件超级列表框.现行选中项 ≤ 0)
    返回 ()
    如果真 (本地文件超级列表框.取表项数值 (本地文件超级列表框.现行选中项) = 1)
        欲上传文件名 = 本地文件超级列表框.取标题 (本地文件超级列表框.现行选中项, 0)
        配置信息.用户名 = 连接窗口.用户名编辑框.内容
        配置信息.用户密码 = 连接窗口.密码编辑框.内容
        配置信息.本地文件路径 = 本地路径编辑框.内容 + 欲上传文件名
        配置信息.传送类型 = 1
        配置信息.类型 = #传送常量.上传 + #传送常量.被动选择
        配置信息.文件地址 = 取文本左边 (连接窗口.地址编辑框.内容,
        取文本长度 (连接窗口.地址编辑框.内容) - 1) + " " + 连接窗口.端口编辑框.内容 +
        远程路径编辑框.内容 + 欲上传文件名
        配置信息.日志回调函数 = @上传日志回调函数
        配置信息.交互回调函数 = @交互日志回调函数
        被上传文件大小 = 到数值 (本地文件超级列表框.取标题 (本地文件超级列表框.现行选中项, 1))
        如果真 (FTP上传对象.上传文件 (配置信息) = 真)
            如果 (设置窗口.上传限制选择框.选中 = 真)
                FTP上传对象.限制速度 (到数值 (设置窗口.上传限制编辑框.内容))
                FTP上传对象.限制速度 (0)

            上传速度时钟.时钟周期 = 1000
            速度透明标签.可视 = 真
            进度条.可视 = 真
```

图 13-48 “上传文件”方法

【注意】本例程只能同时上传一个文件,如果让程序同时可以上传多个文件,则需要一个“FTP 上传对象”数组,数组中每个成员可以开始一项上传任务,但是一个“FTP 上传对象”只能进行一个上传任务。实现多个上传任务的方法和“下载对象”例程实现多个下载任务的方法是相同的。

2. “限制速度()”方法

该方法可以限制上传的速度。上传开始前,上传速度可以在任务参数中进行设置,上传开始后,可以使用本方法限制速度。本方法只有一个整数型参数“速度”,填写欲限制的速度值。

3. “取上传速度()”、“取已上传长()”、“取重试数()”方法

这三个方法用来取得上传过程中的相关信息,从方法名便可知道其作用,这里不做详细介绍。三种方法都没有参数,返回值便是取得的结果。在程序中,可以使用时钟组件,指定周期内在超级列表框或其他组件中刷新显示这 3 个上传相关的信息。

4. “设新文件名()”方法

该方法在上传过程中,更改服务器端正在上传文件的文件名。在参数中填写欲更改的文件名。注意该方法的使用非常特殊,必须是在特定的情况中使用,否则就会上传失败。

首先,该方法必须在上传任务开始前,将“任务参数”的“类型”设置为“#传送常量. 被动选择”;

其次,该方法应该在“交互回调函数”收到的“消息类型”为“#传送常量. 发现存在文件”时使用;

最后,使用完该方法后,必须在“交互回调函数”中返回“#传送常量. 改名”。

5. “取对象信息()”方法

该方法和“下载对象”中的“取对象信息()”方法作用相同,可以取得一个代表当前上传对象的数值。可以在“日志回调函数”和“交互回调函数”中,使用“取对象信息()”方法返回的数值与这两个函数的第五个参数比较,可以了解当前的信息是哪个上传对象返回的。

本节中引用的代码参见例程 13-6。

13.7 邮件接收支持库

邮件接收支持库,可以实现邮件的接收,可以方便地查看接收邮件的各种信息,如邮件标题、发信方邮箱地址和名称等。

13.7.1 接收邮件的基本命令

邮件接收支持库提供了 11 个接收邮件相关的命令,包括:“连接收信服务器()”、“断开收信服务器()”、“获取邮件信息()”、“获取邮件大小()”、“接收邮件()”、“取邮件错误信息()”、“删除邮件()”、“复位邮件()”、“接收邮件序号()”、“接收邮件前几行()”、“注册邮件接收回调函数()”。下面着重介绍一些部分重要命令。

1. “连接收信服务器()”、“断开收信服务器()”命令

“连接收信服务器()”用来连接指定的收信邮箱,其他的命令都是在首先连接服务器后使用的。

语法:〈逻辑型〉连接收信服务器(收信邮件服务器地址,端口号,用户名,密码,最长等待时间,重试次数)

参数：“收信邮件服务器地址”文本型，指定用作接收邮件的 POP3 邮件服务器地址；“端口号”整数型，初始值为“110”，指定欲连接到 POP3 邮件服务器上的端口号，如果本参数被省略，默认值为标准 110 端口；“用户名”文本型，指定在 POP3 邮件服务器上的用户账号名称；“密码”必需的文本型，指定在 POP3 邮件服务器上的用户账号密码；“最长等待时间”整数型，初始值为“30000”，指定在收信过程中等待 POP3 邮件服务器响应的最大时间，单位为毫秒；“重试次数”整数型，初始值为“3”，指定在验证阶段，如果某一项（如：用户名或密码）验证失败后重试的次数，如果本参数被省略，默认值为 3 次。

“断开收信服务器()”命令，用来断开与当前邮件服务器的连接。该命令没有参数和返回值。

语法：无返回值 断开收信服务器()

例如：例题中用到的“连接收信服务器()”方法代码如图 13-49 所示。

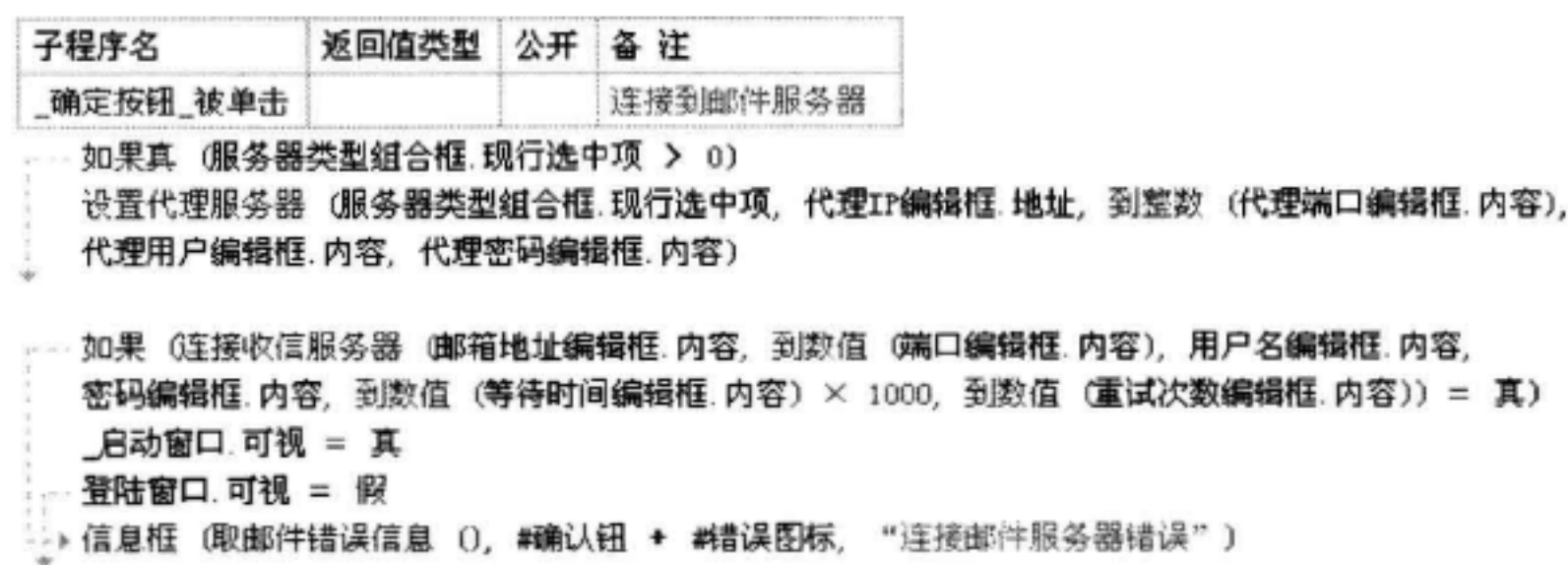


图 13-49 “连接收信服务器”方法

2. “获取邮件信息()”、“获取邮件大小()”命令

“获取邮件信息()”命令可以获取目前服务器上邮件的个数和大小。该命令有 2 个参数：“个数”和“大小”，为这两个参数提供 2 个整数型变量，用来保存获得的结果。如果要使用“计次循环首()”命令来循环收取每一封邮件时，可以使用本命令取得的邮件个数作为循环的次数。

“获取邮件大小()”命令可以获得每一封邮件的大小。

语法：〈逻辑型〉获取邮件大小(第几封, 大小, 总数)

参数：“第几封”整数型，指定欲收取第几封邮件，如果本参数为 -1，则第二个参数可以提供一个数组，保存每一封邮件的大小；“大小”整数型，提供变量或变量数组，如果第一个参数指定第几封邮件，则本参数可以提供整数型变量，来保存取得的这封邮件的大小，如果第一个参数为 -1，则本参数提供一个整数型数组，用来保存每一封邮件的大小；“总数”整数型，本参数在“第几封”参数为 -1 时有效，为“获取邮件信息”方法返回的邮件总数，提供一个整数型变量保存取得的结果。

例程中用到“获取邮件信息()”、“获取邮件大小()”方法的代码如图 13-50 所示。

程序运行后，首先使用取得的邮件个数来重定义“邮件信息”数组的成员数，然后用计次循环收取每一封邮件，并且使用“获取邮件大小()”方法，获取每一封邮件的大小，并保存在数组变量中。

3. “接收邮件()”命令

该命令用来收取邮件的实际内容，返回“邮件信息”类型的数据，提供“邮件信息”类型的变量来保存取得的邮件信息，相当于创建“邮件信息”对象。“邮件信息”对象又提供了相关的方法，取得每一封邮件的详细内容。

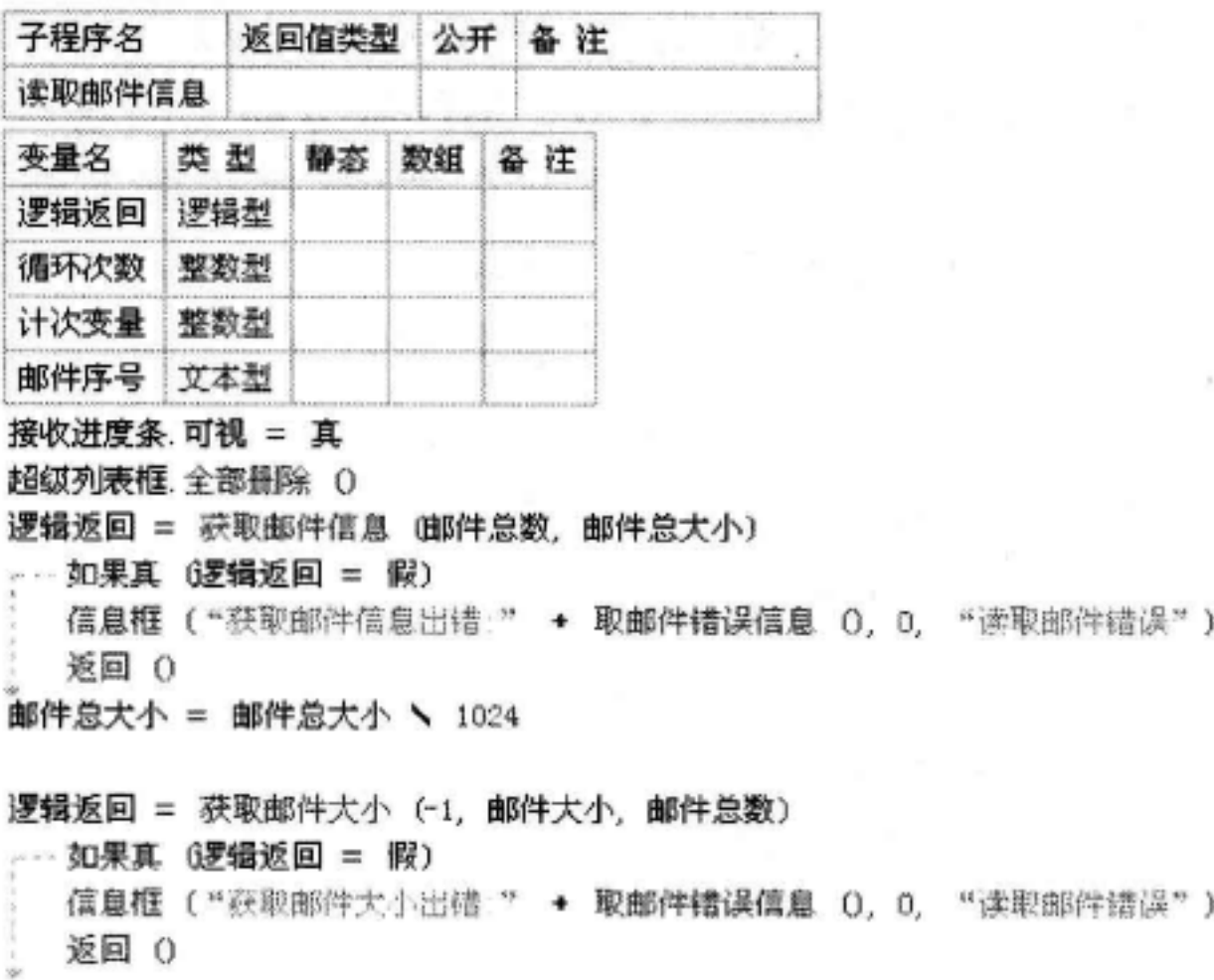


图 13-50 “获取邮件”方法

该命令有一个参数,整数型,指定欲收取第几封邮件。

4. “删除邮件()”、“复位邮件()”命令

“删除邮件()”命令可以删除邮箱中的指定邮件,注意该命令是在邮件服务器端删除邮件,而不是在本地收取的邮件中删除。本命令有一个整数型参数,指定欲删除第几封邮件。例如,例程中用到“删除邮件()”命令的代码如图 13-51 所示。

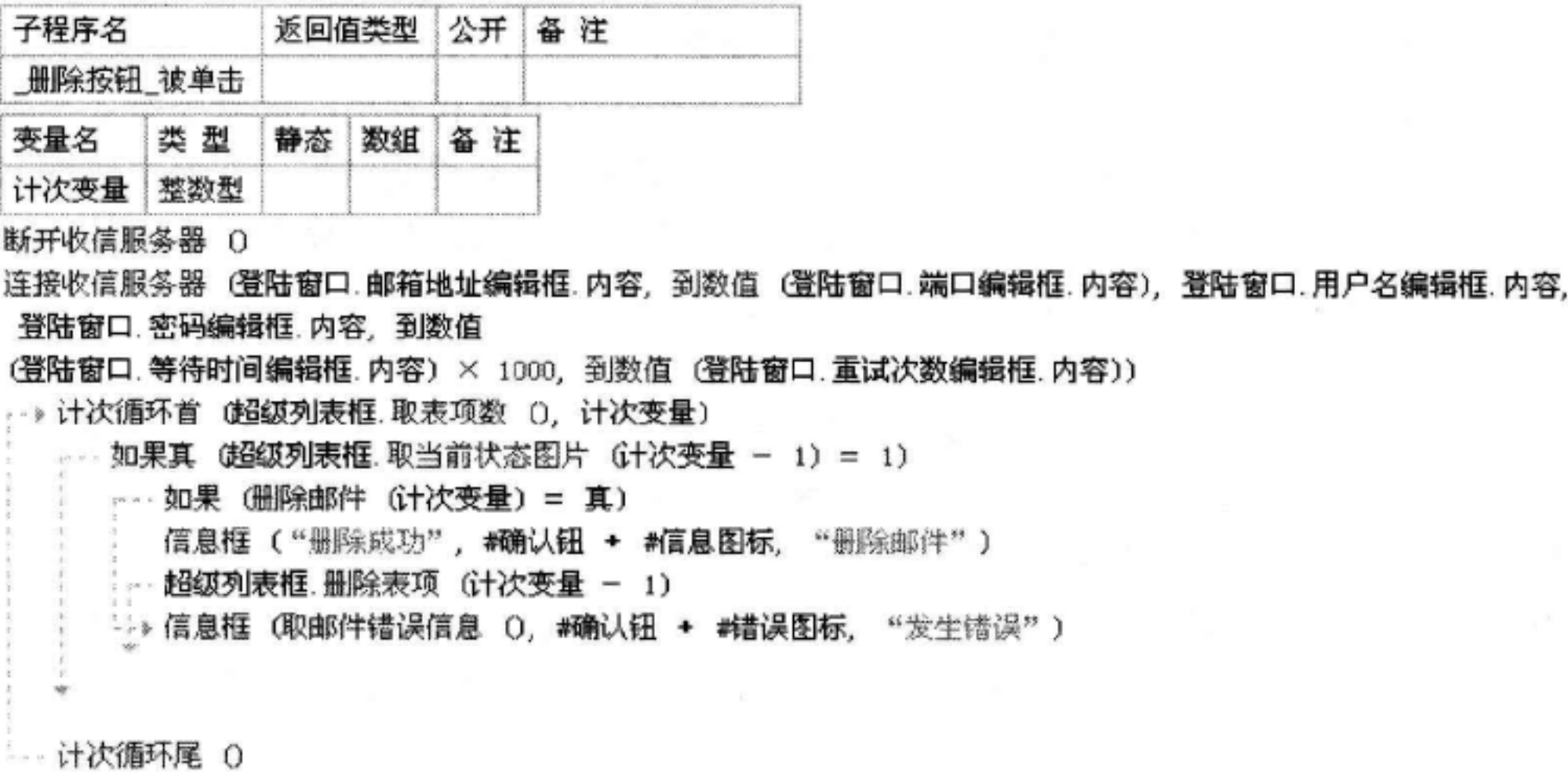


图 13-51 “删除邮件”方法

“复位邮件()”命令用来将服务器上标记为删除的邮件标记为未删除。

5. “接收邮件序号()”命令

可以取得邮件在服务器中的唯一序号,此序号对于每个服务器是唯一的,可用来比较邮件是否接收到本地。

该序号可以用来判断是否已经接收过某邮件,例如曾接收过的邮件在本地机器删除后,下次不想接收了,就可以保存这个序号,下次不接收该序号的邮件即可。

6. “注册邮件接收回调函数()”命令

注册一个回调函数,主要用于接收邮件时的进度显示。

【注意】此回调函数中不要进行耗时的操作,否则邮件服务器会断开!不要调用邮件接收支持库中和服务器交互的命令,否则可能出现死循环!

该命令有一个子程序指针型参数,参数中填写回调函数的指针。

邮件接收回调函数的参数定义如下:

第一个参数为整数型,表示接收邮件命令类型值,是“命令类型”枚举常量中的一个值,大家可以参照易语言知识库中,对该常量值的解释;第二个参数为整数型,表示当前正在接收第几封邮件;第三个参数为整数型,表示本次接收到的数据量单位为字节。子程序返回值为逻辑型,返回“真”表示继续接收此邮件,返回“假”表示不接收此邮件。

13.7.2 “邮件信息”数据类型的方法

“接收邮件()”命令取得的是“邮件信息”类型的数据,如果要取得每一封邮件的详细信息和内容,就要通过“邮件信息”对象提供的方法来实现。

1. “取发件人地址()”、“取主题()”、“取日期()”、“取文本内容()”、“取附件个数()”、“取发件人名称()”、“取回复地址()”方法

以上方法都是取得邮件的基本信息,都没有参数,返回取得的结果。

- “取发件人地址()”方法取得发信人的地址。
- “取主题()”方法返回邮件主题。
- “取日期()”方法返回发信日期,返回值为日期时间型数据。
- “取文本内容()”方法返回邮件内容,即信件的正文部分。
- “取附件个数()”方法返回当前邮件的附件个数。
- “取发件人名称()”方法取得发信人的邮箱地址。
- “取回复地址()”方法返回发信人指定的回复地址,如果发信人没有指定回复地址,则默认为发信人的邮箱地址。

例程中用到“取发件人地址()”、“取主题()”、“取日期()”、“取文本内容()”、“取附件个数()”、“取发件人名称()”、“取回复地址()”方法的代码如图13-52所示。

```

--> 变量循环首 (1, 邮件总数, 1, 循环次数)
  处理事件 0
  接收邮件序号 (循环次数, 邮件序号)
  置当前库 (邮件数据库名)
  如果 (是否快速浏览 = 假)
    邮件 = 接收邮件 (循环次数)
    邮件 = 接收邮件前几行 (循环次数, 10)
  加记录 (邮件.取发件人名称 0, 邮件.取主题 0, 邮件.取日期 0, 邮件.取发件人地址 0, 邮件.取回复地址 0,
  邮件.取附件个数 0
  邮件.取超文本内容 0, 邮件.取原始信息 0; 邮件大小 [循环次数], 邮件序号)
  显示邮件信息 0
  置当前库 (附件数据库名)
  附件 = 邮件.取附件 0
  --> 计次循环首 (邮件.取附件个数 0, 计次变量)
    加记录 (选择 (附件 [计次变量].取是否嵌入式附件 0, 附件 [计次变量].取名称 0, 附件 [计次变量].取文件名 0),
    附件 [计次变量].取类型 0附件 [计次变量].取是否嵌入式附件 0, 附件 [计次变量].取数据 0, 邮件序号)
  -- 计次循环尾 0
-- 变量循环尾 0

```

图 13-52 “删除邮件”方法

2. “取数据()”方法

该方法取得附件数据,可以用来将附件保存到文件。

3. “取是否嵌入式附件()”方法

该方法用来判断当前附件是否是嵌入式附件,如网页上的图片,就作为嵌入式附件下载到本地。

本节中用到的代码参见例程 13-7。

13.8 局域网操作支持库

局域网操作支持库,用来浏览局域网中所有主机的共享目录,并可以将指定的共享目录映射为网络驱动器。映射后的共享目录,会在“我的电脑”中显示为一个驱动器盘符,点击该盘符,就会自动访问被映射的共享目录,使用网络映射可以为访问局域资源提供方便。

局域网操作基本命令

局域网操作支持库中包含一个“局域网操作”对象,该对象提供的方法用来对局域网资源进行操作,所以在使用本支持库命令前,先要创建一个“局域网操作”类型的变量,即创建一个“局域网操作”对象。下面就来对该对象提供的方法进行介绍。

1. “取共享资源()”、“取所有主机名()”方法

“取共享资源()”方法,取得局域网中的所有共享资源,将取得的资源存放在参数中提供的文本型数组变量中。

语法:〈整数型〉局域网操作.取共享资源(资源类型 欲取的资源类型,文本型变量数组 取得的资源,[整数型变量数组 取得的资源类型])

参数:“欲取的资源类型”该参数可以为“资源类型”枚举常量集合中的一个常量值;“取得的资源”文本型,提供参数数据时只能提供变量数组。命令执行完毕后本变量数组中依次存放取得的资源的路径;“取得的资源类型”整数型,可以省略,提供参数数据时只能提供变量数组,命令执行完毕后本变量数组中依次存放取得的资源的类型,并与取得的资源路径一一对应。

“取所有主机名()”方法,取得局域网中的所有连接主机的名称。将取得的名称存放在参数中提供的文本型数组中。该方法只有一个参数,参数中填写用来存放所有主机名的数组变量。如图 13-53 所示。

子程序名	返回值类型	公开	备注
取共享打印机			

局域网.取共享资源 (#资源类型.共享打印机,共享打印机,)
取共享打印机结束 = 真

子程序名	返回值类型	公开	备注
取共享文件夹			

局域网.取共享资源 (#资源类型.共享文件夹,共享文件夹,)
取共享文件夹结束 = 真

子程序名	返回值类型	公开	备注
取所有主机名			

局域网.取所有主机名 (主机名)
取主机名结束 = 真

图 13-53 “取所有主机名”方法

【注意】“取所有主机名()”和“取共享资源()”方法运行都需要一定的时间,因为要与其他主机连接,在启动窗口创建后,会有一段未响应时间。

代码中,将所有主机名取得后,将所有的主机名添加到组合框中,方便以后查找指定主机的共享资源时选择。

2. “取连接速度()”方法

该方法取出与局域网内指定的共享资源(共享目录或共享打印机等)之间的连接速度。以“字节/秒”为单位。如果该命令执行出错,返回-1;如果无法取出连接速度,返回0。

【注意】该命令取出的速度为大概值,可能有误差,仅供参考。该方法有一个参数,为“欲测试的资源路径”,文本型,填写局域网中的共享文件夹路径。

例程中用到“取连接速度()”方法的代码如图13-54所示。

子程序名	返回值类型	公开	备注
_取连接速度_被选择			

变量名	类型	静态	数组	备注
返回值	整数型			
速度	双精度小数型			

返回值 = 局域网2.取连接速度 (共享文件夹列表框.取项目文本 (共享文件夹列表框.现行选中项))

判断 (返回值 = -1)

信息框 (“无法取出与该共享的连接速度!” + #换行符 + “错误信息为:” + 局域网2.取错误信息 0, #错误图标,)

判断 (返回值 ≠ 0)

速度 = 返回值

速度 = 速度 ÷ 1024

信息框 (“与该共享的连接速度大概为:” + 到文本 (速度) + “KB/s”, #信息图标,)

信息框 (“暂时无法取得与该共享的连接速度!”, #信息图标,)

图13-54 “取连接速度”方法

3. “映射资源()”、“断开映射()”方法

“映射资源()”方法,映射一个局域网内的共享资源到本地驱动器。成功返回映射到本地的驱动器名称,失败返回空文本。

语法:〈文本型〉局域网操作.映射资源(欲映射的资源路径,[欲映射到的本地设备名称],[用户名],[密码],[是否提示错误],[是否自动重新连接])

参数:“欲映射的资源路径”文本型。该参数可以为局域网中的共享文件夹或共享打印机的路径,比如“\\Prog5\共享测试”,“\\Prog5\惠普打印机”;“欲映射到的本地设备名称”文本型。比如“Y:”,“LPT1”。如果本参数省略,系统将自动选择一个可用的设备名称;“用户名”文本型。访问指定的共享使用的用户名,如果本参数省略,将使用系统默认值;“密码”文本型,访问指定的共享使用的密码,如果本参数省略,将使用系统默认值;“是否提示错误”逻辑型,本参数指定当用户名或密码错误时,是否提示重新输入,如果提供的用户名或密码有误,且本参数为“假”,该命令将失败;“是否自动重新连接”逻辑型,本参数指定是否在下次登陆计算机时自动重新连接该映射驱动器,如果本参数省略,默认值为“真”。

“断开映射()”方法,断开一个映射到本地驱动器的共享资源。成功返回“真”,失败返回“假”。

语法:〈逻辑型〉局域网操作.断开映射(欲断开的映射驱动器,是否永久断开,是否强制断开)

参数：“欲断开的映射驱动器”文本型。本参数指定欲断开连接的映射驱动器（或打印机）名称。比如“X:”或“LPT1”；“是否永久断开”逻辑型，初始值为“真”，本参数指定是否永久断开该映射驱动器，永久断开的映射驱动器在下次登录时不会自动重新连接；“是否强制断开”逻辑型，初始值为“假”，如果本参数为“真”，即使本映射驱动器正在使用，也立即强制断开。

例程中用到“映射资源()”、“断开映射()”方法的代码如图 13-55、图 13-56 所示。

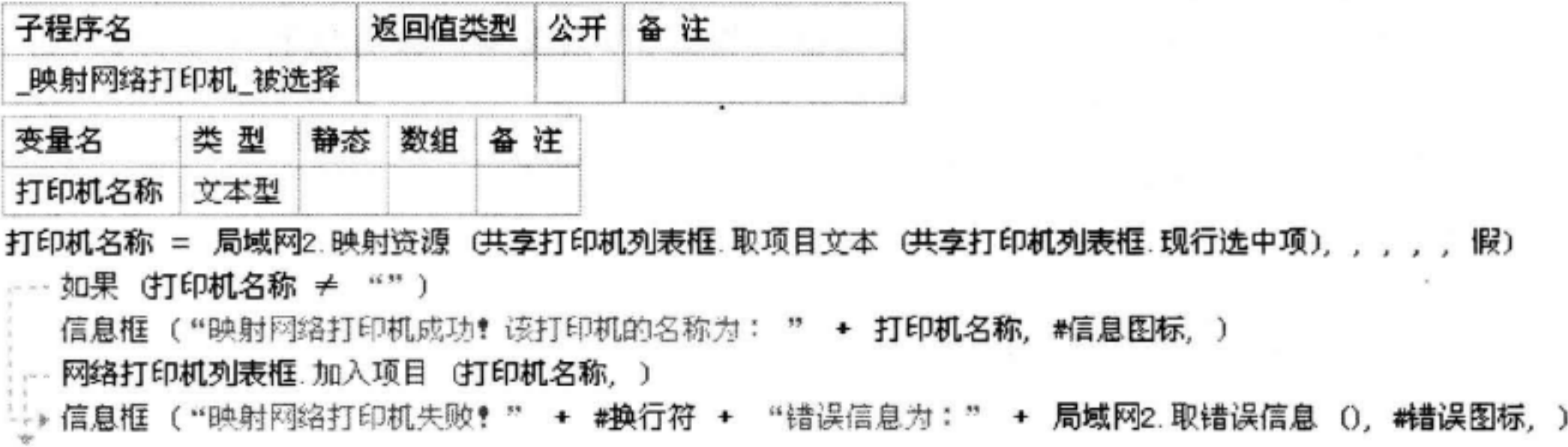


图 13-55 “映射资源”方法

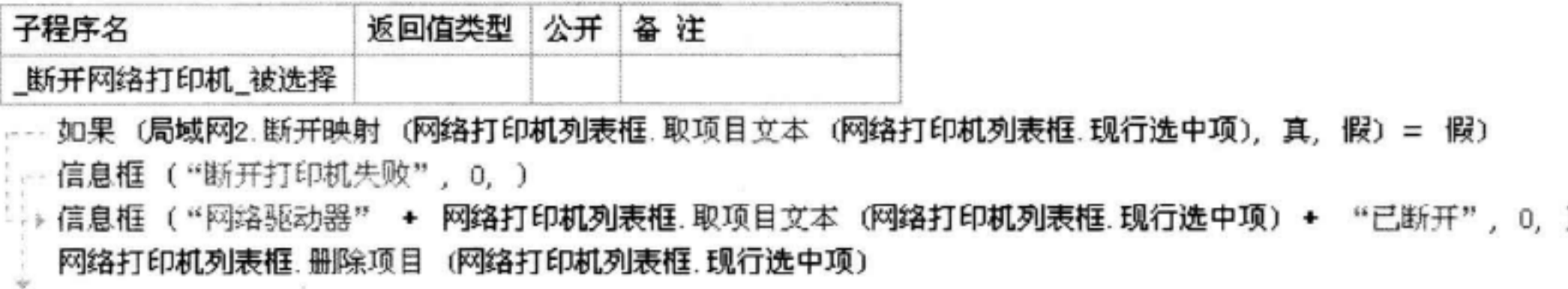


图 13-56 “断开映射”方法

4. “打开映射对话框()”、“打开中断对话框()”方法

这两个方法都是用来打开系统自带的对话框。

“打开映射对话框()”方法，打开系统“映射网络驱动器”对话框。如果通过该对话框成功映射了一个网络驱动器，返回该驱动器的名称，否则返回空文本。

语法：〈文本型〉局域网操作. 打开映射对话框(网络路径初始值,网络路径可否输入,重新连接是否可选,是否自动连接,[父窗口句柄])

参数：“网络路径初始值”文本型，初始值为空文本。本参数指定出现在“映射网络驱动器”对话框中网络路径编辑框中的初始值；“网络路径可否输入”逻辑型，初始值为“真”，本参数指定对话框中的网络路径编辑框是否可用；“重新连接是否可选”逻辑型，初始值为“真”，本参数指定对话框中“登录时重新连接”复选框是否可用；“是否自动连接”逻辑型，初始值为“真”，本参数指定对话框中“登录时重新连接”复选框是否被选中；“父窗口句柄”整数型，本参数指定打开的对话框的父窗口。

例程中用到“打开映射对话框()”代码如图 13-57 所示。

“打开中断对话框()”方法用来打开系统“中断网络驱动器”对话框。该方法只有一个参数，为“父窗口句柄”填写作为本对话框的父窗口的句柄。例程中用到“打开中断对话框()”代码如图 13-58 所示。

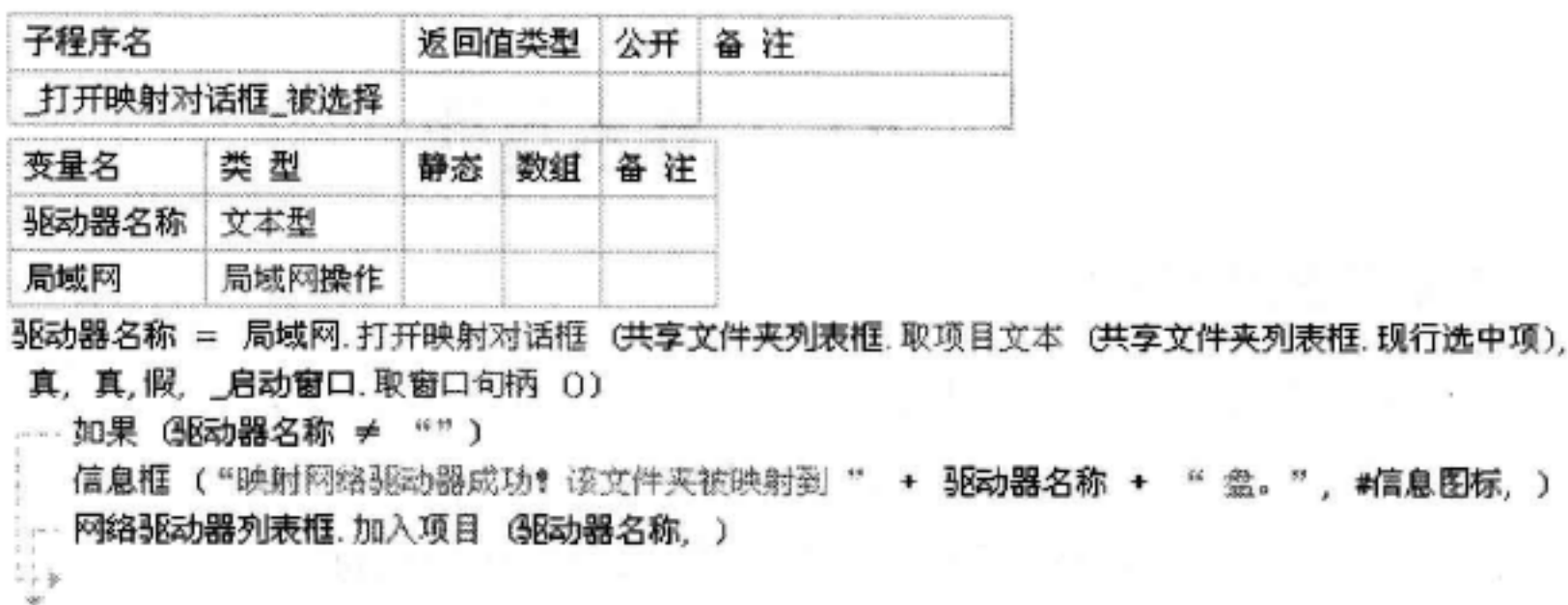


图 13-57 “打开映射对话框”方法

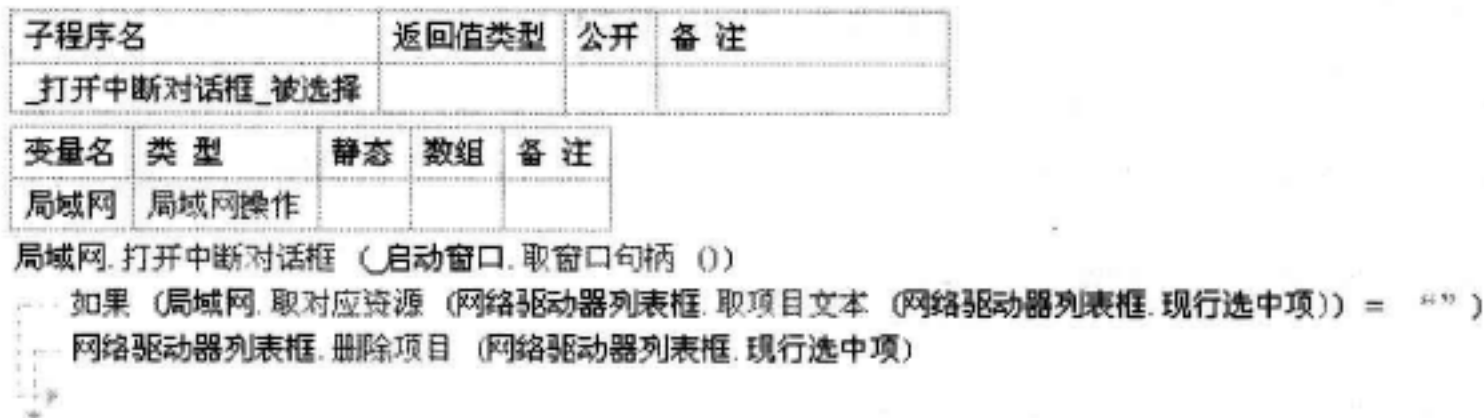


图 13-58 “打开中断对话框”方法

5. “取对应资源()”、“取资源路径()”、“取错误信息()”方法

“取对应资源()”取得指定的网络驱动器所对应的共享资源路径。失败返回空文本。该方法只有一个参数,参数中填写欲取得对应资源的映射驱动器名称。

“取资源路径()”取得指定的网络驱动器的路径所对应的局域网资源路径、连接名称和相对路径。

语法:〈逻辑型〉局域网操作.取资源路径(网络驱动器路径,取得的资源路径,[取得的连接名称],[取得的相对路径])

参数:“网络驱动器路径”文本型,填写欲取得相关信息资源名称;“取得的资源路径”提供一个文本变量,存放取得的资源路径;“取得的连接名称”提供一个文本变量,存放当前共享资源的连接名称;“取得的相对路径”提供一个文本变量,取得的相对路径。例程中用到“取资源路径”方法的代码如图 13-59 所示。

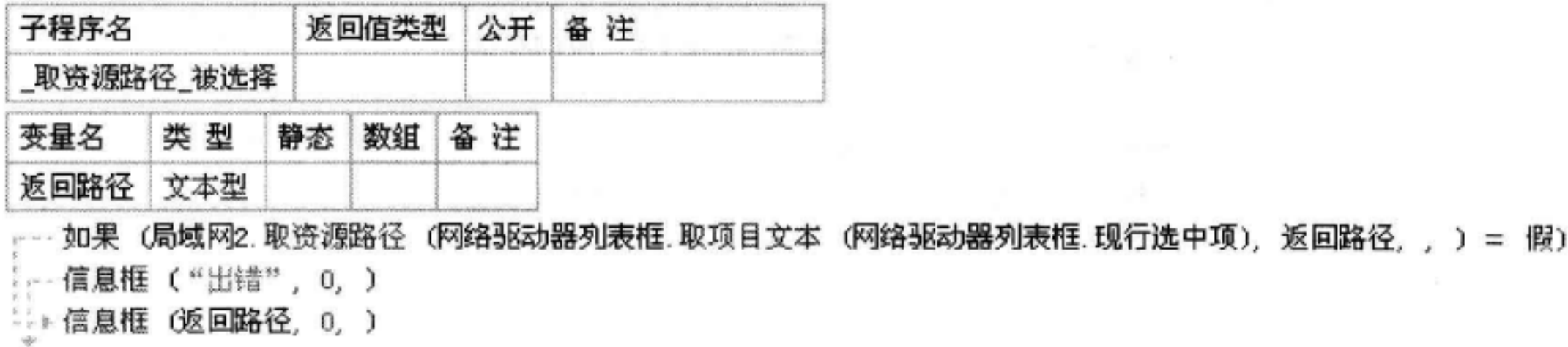


图 13-59 “打开中断对话框”方法

“取错误信息()”方法,当调用“局域网操作”的成员命令失败后,可以立即调用该命令来取出相关的错误信息。注意:请仅在调用“局域网操作”的成员命令失败后调用本命令,否则取得的信息没有意义。该方法没有参数,直接返回取得的错误信息文本。

本节应用的代码参见例程 13-8。

13.9 BT 下载支持库

13.9.1 BT 下载支持库

随着 Internet 网络的普及,网络上下载资料、音乐等等已经非常流行,所以各种下载方式也随之产生。BT 下载以其独特的下载方式,很快赢得了网友们的青睐,迅速成为了网络上主要的下载方式之一,BT 下载支持库可以方便大家实现 BT 下载功能。

BT 下载需要一个记录了服务器地址、文件地址、文件名称等等一系列详细内容的种子文件来实现下载(种子文件扩展名为 Torrent),在各大 BT 下载网站上都发布了大量的种子,只需要使用 BT 下载软件打开该种子文件,便可以开始下载。

BT 支持库提供了 5 个全局命令,和一个“BT 下载”数据类型(即用来创建 BT 下载对象)。全局命令用来对 BT 支持库整体进行设置,BT 下载对象提供的方法,用来进行 BT 下载任务。

13.9.2 BT 下载支持库全局命令

1. “分析发布文件()”、“制做发布文件()”命令

“分析发布文件()”命令,可以取得 BT 种子文件中的相关信息,返回“发布文件信息”类型的数据,该类型数据包括了服务器地址、文件地址、文件名称等成员,是本支持库中自定义的数据结构类型,大家可以打开易语言支持库手册来查看该类型数据都包括哪些成员。本命令只有一个参数,填写种子文件在本地机器上的路径名。

该命令取得的 BT 种子中的相关信息,主要用于在软件中显示这些下载的相关信息。

“制做发布文件()”用来给本地欲共享的文件制作下载种子文件,可以在网站上发布,提供给其他人下载共享的文件。

语法:〈整数型〉制做发布文件(文件类型,名称,服务器地址,发布路径,块大小,注释)

参数:“文件类型”整数型,初始值为“2”,本参数表示要发布文件的类型 1 表示目录,2 表示文件;“名称”文本型,本参数表示目录名或文件名;“服务器地址”文本型。本参数表示 Tracker 服务器的地址。多个服务器之间用“;”号分隔;“发布路径”文本型。本参数表示生成的发布文件的路径;“块大小”整数型,本参数表示每一块的大小,值可以为“块大小”数据类型中的常量值;“注释”文本型,初始值为“”。本参数表示关于此发布文件的注释。

例程中用到“分析发布文件()”命令的代码如图 13-60 所示。

子程序名	返回值类型	公开	备注
分析发布文件子程序			分析发布文件的内容

文件信息 = 分析发布文件(发布文件的名称)

图 13-60 “分析发布文件”命令

2. “下载设置()”命令

“下载设置()”命令用来对 BT 下载支持库的全局参数进行设置。该命令有一个参数,为“下载设置信息类型”。程序中可以定义一个该类型的变量,然后对该类型变量的每一个成员进行赋值,最后在“下载设置()”命令的参数中填写设置完成的变量。例如,例程中用到“下载设置()”命令的代码如图 13-61 所示。

子程序名	返回值类型	公开	备 注	
_设置按钮_被单击				
变量名	类 型	静态	数组	备 注
下载设置信息	下载设置信息			
下载设置信息.每地址连接数 = 到数值 (每地址连接数编辑框.内容)				
下载设置信息.开始监听端口 = 到数值 (开始监听端口编辑框.内容)				
下载设置信息.结束监听端口 = 到数值 (结束监听端口编辑框.内容)				
下载设置信息.阻塞值 = 到数值 (阻塞值编辑框.内容)				
下载设置信息.连接超时 = 到数值 (连接超时编辑框.内容)				
下载设置信息.代理服务器类型 = 代理服务器类型组合框.取项目数值 (代理服务器类型组合框.现行选中项)				
下载设置信息.代理服务器地址 = 代理地址编辑框.内容				
下载设置信息.代理服务器端口 = 到数值 (代理端口编辑框.内容)				
下载设置信息.用户名 = 编用户名.内容				
下载设置信息.口令 = 编口令.内容				
下载设置 (下载设置信息)				

图 13-61 “下载设置”命令

3. “重新检查完整性()”、“测试代理服务器()”命令

“重新检查完整性()”命令用来重新检查下载后文件的完整性。

语法:〈逻辑型〉重新检查完整性(发布文件名,本地文件路径,本地文件名,百分比,字节数,任务内容,日志)

参数:“发布文件名”文本型,本参数表示要检查的种子文件名(torrent);“本地文件路径”文本型,本参数表示被下载文件的本地保存路径;“本地文件名”文本型,初始值为“”,本参数表示被下载文件的本地文件名,如本参数为空则用发布文件中默认的文件名;“百分比”整数型,参数数据只能提供变量,本参数传回下载了百分之几;“字节数”长整数型,参数数据只能提供变量,本参数传回已写入硬盘的字节数;“任务内容”字节集,参数数据只能提供变量,本参数传回任务内容,可用传入“增加新任务”方法的参数中,用来避免效验文件;“日志”子程序指针,本参数表示检查进度的日志回调函数,详见“其他日志”数据类型的“检查完整性中”成员。例如,例程中用到“重新检查完整性()”命令的代码如图 13-62 所示。

子程序名	返回值类型	公开	备 注	
检查完整性			重新检查下载后文件的完整性	
变量名	类 型	静态	数组	备 注
百分比	整数型			
字节数	长整数型			
任务内容	字节集			
函数返回值	逻辑型			
文件号	整数型			

函数返回值 = 重新检查完整性 (当前下载任务信息.发布文件名,保存路径, “”, 百分比, 字节数, 任务内容, &其它日志)

图 13-62 “重新检查完整性”命令

13.9.3 “BT 下载”数据类型的提供的方法

“BT 下载”提供的方法有“增加新任务()”、“暂停本任务()”、“继续本任务()”及“停止本任务()”四种方法。

“增加新任务()”方法用于增加一个下载任务。

【注意】本函数返回真并不是已经真正开始下载了,要通过“其他日志”中的“下载已全部停止”的参数三来判断。本方法有一个参数,为“任务信息”类型,程序中需要定义一个“任务信息”类型的变量,然后设置该类型变量每个成员的值,然后将此变量赋值给“增加新任务()”的参数。

例程中用到增加本任务、下载本任务信息的代码如图 13-63 所示。

子程序名	返回值类型	公开	备注
下载进行中			

变量名	类型	静态	数组	备注
任务内容	字节集			

本任务信息.发布文件名 = 种子路径名称
本任务信息.本地文件路径 = 保存路径
本任务信息.主动连接数 = 1
本任务信息.最大连接数 = 100
本任务信息.任务内容 = 任务内容
当前下载.增加新任务(本任务信息)

子程序名	返回值类型	公开	备注
_停止按钮_被单击			

当前下载.停止本任务()

图 13-63 “增加新任务”方法

“暂停本任务()”、“继续本任务()”、“停止本任务()”方法与“增加新任务()”方法相类似,在此就不再介绍。

本节中引用的代码参见例程 13-9。

第 14 章 子程序调用

目前软件系统设计常用的方法就是采用模块化结构,使得系统结构清晰、易读、易维护,模块化结构程序设计方法就是指将一个个具有特定功能的程序段构造为若干个独立的模块,再把它们按照要实现的功能进行不同的组合,构成一个完整的程序来实现特定的任务。

子程序就是模块化结构设计中的重要组成,通过对一系列命令进行封装,实现模块化、重用及抽象,从而有利于程序的结构化开发,使程序结构更加清晰。因此,我们将程序分割成较小的逻辑单元以便简化程序设计任务,这些逻辑单元就被称为子程序。子程序可用于压缩重复任务或共享任务,例如,压缩频繁的计算处理等等。

子程序相对主程序来存在的,它具有程序的一般特点,如都是由基本的命令组成,能够实现一个特定的功能等,它是系统程序的组成部分,不能离开系统程序而独立存在,但是可以将甲程序编制好的一个子程序拷贝到乙程序中来使用,节省了为实现同一功能在乙程序中重新编写代码的工作。

简单地说,子程序就像建筑工地上常用的一个个工件,楼板、砖(空心砖)、水泥、钢筋、门、窗等,要按照图纸设计进行搭建,就可以搭出各种各样的建筑来。通过子程序的使用,可以避免对相同代码多次重复编写的麻烦,也可以减少出错的机会。

◎ 子程序可使程序划分成离散的逻辑单元,每个单元都比无子程序的整个程序容易调试及理解。

◎ 一个应用程序中的子程序,往往不必修改或只需稍作改动,便可以成为另一个程序的子程序。

◎ 在子程序中对代码的修改,能够影响到所有调用该子程序的程序功能实现。

◎ 将一个相对复杂的功能,分解为多个简单的功能,通过子程序逐个实现,有利于提高代码的正确性和清晰度,减少代码编写出错的可能性。

14.1 子程序的分类及建立

14.1.1 子程序的分类

易语言中的子程序可以分为两大类:“事件子程序”和“用户自定义子程序”。

易语言提供了大量的程序组件,或者控件,是一个个独立的功能模块。把多个控件有序组合起来,就构成了一个程序。这些控件都有一个“事件”属性,对应组件所发生事件子程序,称作组件的“事件子程序”。

例如:按钮有被单击、被双击、左键被按下等事件,选中按钮后,在属性面板中的事件列表中选择这些事件,就可以生成对应的事件子程序。事件的产生由易语言系统全权负责。当某一事件产生时,系统将自动调用该事件所对应的“事件处理子程序”。编程者所要做的,只是根据需求选择要响应的事件,并在该事件的“事件处理子程序”中输入代码即可。运行时,一

且这些事件产生,就会自动执行相应的子程序。事件子程序的名称、返回值类型和参数个数都是系统定义的,不允许用户任意修改。

“用户自定义子程序”是由用户使用易语言环境提供的命令自己创建的子程序,其参数个数和返回值都由用户自行定义的,用户可以根据需要在程序设计时对其任意修改。

14.1.2 子程序的建立

子程序是存在于程序集中的,由程序集分组管理。一个新建的易程序还没有进行任何操作是没有程序集的。有三种方法来创建程序集,一是在“窗口”控件的空白处双击鼠标,会自动创建该窗口对应的窗口程序集,并自动生成“_启动窗口_创建完毕”子程序(创建完毕事件子程序可包含本窗口创建时被执行的程序代码),二是双击窗口中“按钮”等组件,也会创建该窗口对应的窗口程序集及组件对应缺省事件子程序(如按钮被单击),三是使用菜单“插入”→“程序集”来新建一个程序集。

14.1.2.1 事件子程序的创建

控件的通用“事件”主要有:1. 鼠标左键被按下;2. 鼠标左键被放开;3. 被双击;4. 鼠标右键被按下;5. 鼠标右键被放开;6. 鼠标位置被移动;7. 按下某键;8. 放开某键;9. 字符输入;10. 获得焦点;11. 失去焦点。易语言中的可视控件,都有上述事件。除了这些通用事件,控件一般都还有自己的专有事件。如“按钮”控件的专有事件“被单击”。

事件子程序的创建也比较简单,只要选中相应控件,在其属性框下方有一个选项“在此处选择加入事件子程序”,单击选项,就会弹出相应事件列表,根据编程需要选择相应列表选项即可。下面以建立一个“按钮”控件“被单击”为例进行说明。

(1) 在窗口建立“按钮”控件,并选中“属性框”,如图 14-1 所示。

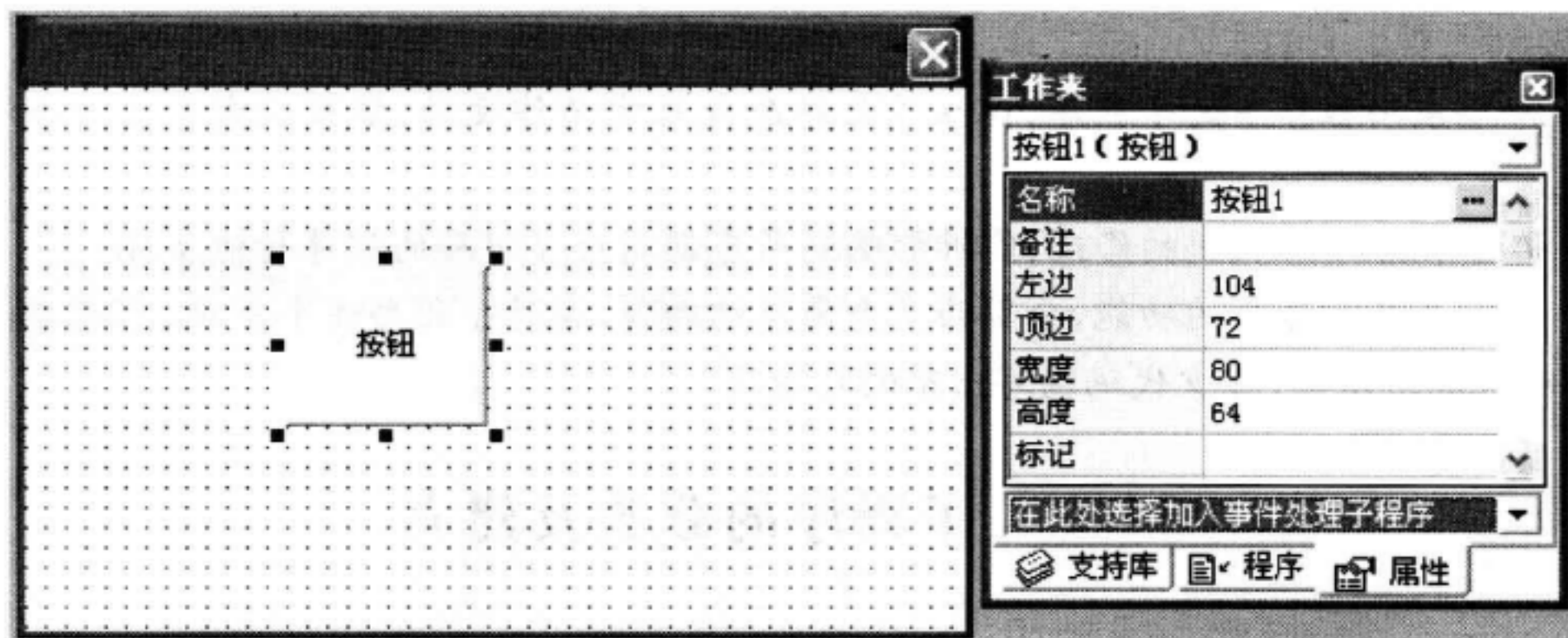


图 14-1 建立控件

(2) 单击“在此处选择加入事件子程序”,弹出事件列表,选中其中“被单击”事件,如图 14-2 所示。

(3) 在窗口对应的程序集“窗口程序集 1”中,可以看到一个名为“_按钮 1_被单击”的事件子程序建立起来了,如图 14-3 所示。

【注意】事件子程序的名称、返回值类型和参数个数都是系统定义的,不允许用户任意修改,否则就会出错。

◎子程序具有“程序集”属性,也就是说,子程序一般只

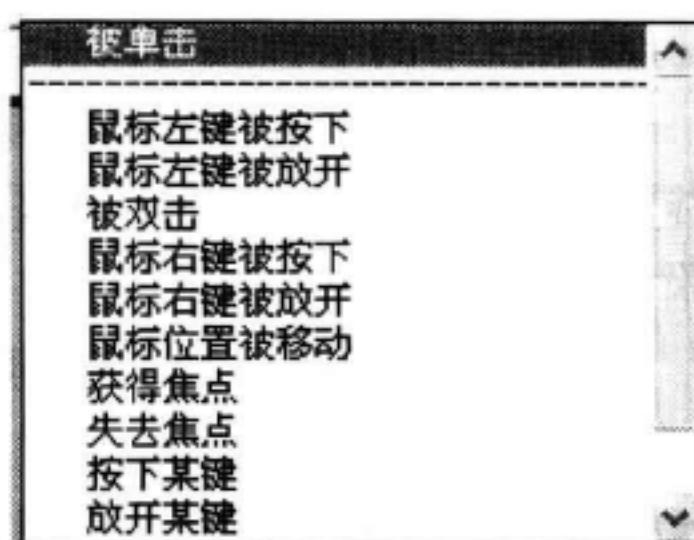


图 14-2 为控件选择事件

窗口程序集名	保留	保留	备注
窗口程序集1			

子程序名	返回值类型	公开	备注
_按钮1_被单击			

图 14-3 建立事件子程序

能在本程序集中被其他程序调用,其他程序集是不能直接调用的。为了便于子程序的应用,子程序提供了一个“公开”属性,用来告知该子程序是否可以被其他程序集调用,如果“公开”属性被“√”勾选,则可以被任何程序集调用。

(4) 现在可以在“_按钮 1_被单击”子程序中添加自己的代码了。如加入“信息框 (“按钮被单击事件发生了!” ,0,“按钮被单击”)”命令,则在单击按钮的时候程序会弹出一个信息提示。如图 14-4 所示。

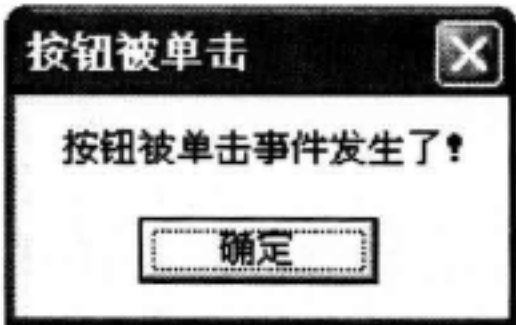


图 14-4 加入信息框的运行效果

14.1.2.2 用户自定义子程序的创建

用户自定义子程序可以放在窗口程序集中,或自定义的程序集中,一般来讲,当前窗口用到的用户自定义子程序应该放在当前窗口程序集中,这样就可以非常方便地引用程序集变量和窗口中的组件,而且查找起来也较为方便。

若程序集已建立,可通过程序面板切换到程序集中操作,或通过窗口菜单切换。

有三种方法来创建一个新的用户自定义子程序。一是在代码编辑面板获得焦点时,在主菜单上选择“插入”→“新子程序”来新建用户子程序,二是在代码编辑面板获得焦点时,右键单击,在弹出菜单中选择“新子程序”命令来新建用户子程序,三是采用更快捷的方法,在代码编辑面板获得焦点时,同时按下“Ctrl”键和“N”键,也可以新建用户子程序。如图 14-5 所示。

窗口程序集名	保留	保留	备注
窗口程序集1			

子程序名	返回值类型	公开	备注
_按钮1_被单击			

子程序名	返回值类型	公开	备注
_启动窗口_创建完毕			

子程序名	返回值类型	公开	备注
子程序1			

N 新子程序Ctrl+N

U 撤消Ctrl+Z

C 复制Ctrl+C

T 剪切Ctrl+X

图 14-5 创建新的自定义子程序

新建的用户子程序缺省名称为“子程序”和一个数字序号的组合,如“子程序 1”,创建后的子程序可以根据需要修改其名称,建议在定义子程序名称时,尽量选择能体现其功能、比较统一规范的名称,如“子程序_求和”、“子程序_统计字符”等,统一归类管理,方便调用时查找。如图 14-6 所示。

◎在同一个程序集中,子程序名不能重复。特别在修改的时候,“易语言”会提示并修改

子程序名	返回值类型	公开	备注		
子程序_求和	整数型				
参数名	类型	参考	可空	数组	备注
加数A	整数型				
加数B	整数型				

子程序名	返回值类型	公开	备注		
子程序_统计字符	整数型				
参数名	类型	参考	可空	数组	备注
字符串	文本型				

子程序名	返回值类型	公开	备注		
_按钮1_被单击					

zcx

☒ 子程序_求和

☒ 注册项是否存在

☒ 子程序_统计字符

图 14-6 定义子程序名称

已经被使用过的名称。

◎新建的子程序没有参数和变量,这需要用户自行添加和自定义参数的数量以及每个参数的类型、可空、参考、数组属性。

14.2 子程序的调用

14.2.1 子程序的调用

子程序的调用方法和命令的调用方法相同,输入子程序的名称就可以调用该子程序。如果子程序要求提供参数,在括号中要按照子程序参数定义的数据类型和参数个数顺序填写参数;如果需要用变量保存子程序返回值,必须使用和子程序返回值相同的数据类型变量。

子程序可以在其他事件子程序和新建子程序中多次被调用,也可以被其他程序集的事件子程序和新建子程序调用。例如单击“按钮 1”,子程序“_按钮 1_被单击”将被调用,“子程序 1”也将被调用执行,这种子程序中再调用另一个子程序的过程,叫子程序嵌套。如图 14-7 所示。

每次调用子程序时,子程序中的所有语句都将被从第一条开始顺序执行,当执行到子程序尾部或者遇到“返回”命令时即返回到调用此子程序语句的下一条语句处。

要注意的是,如果“公开”属性没有被勾选的话,子程序只能在本程序中使用。

14.2.2 子程序的递归调用

一个子程序不仅容许其他子程序来调用,也可以被自身调用,这种调用子程序本身的过程称为子程序的递归调用。

下例中编写了一个循环计数的例子,由于“子程序 1”中调用了本身“子程序 1”,系统能够不停计数显示。具体的程序代码如图 14-8 所示。

子程序名	返回值类型	公开	备注
_按钮1_被单击			
子程序1 0			
子程序名	返回值类型	公开	备注
子程序1			
信息框 (“你好！我是信息框。”, #信息图标,)			
子程序名	返回值类型	公开	备注
_按钮2_被单击			
子程序1 0			

图 14-7 子程序嵌套示例

窗口程序集名	保留	保留	备注
窗口程序集1			
变量名	类型	数组	备注
y	整数型		
子程序名	返回值类型	公开	备注
_启动窗口_创建完毕			
子程序名	返回值类型	公开	备注
_按钮1_被单击			
子程序1 0			
子程序名	返回值类型	公开	备注
子程序1			
y = y + 2			
+ 输出调试文本 (“子程序嵌套运行结果:” + 到文本 (y))			
子程序1 0			

图 14-8 “子程序递归调用”代码

单击“按钮 1”，触发“按钮 1 被单击”事件，执行“_按钮 1_被单击”子程序，其中调用了子程序“子程序 1”，使得调试文本结果不断按增 2 变化。如图 14-9 所示。

提示	输出	调用表	监视表
* 子程序嵌套运行结果: 874			
* 子程序嵌套运行结果: 876			
* 子程序嵌套运行结果: 878			
* 子程序嵌套运行结果: 880			
* 子程序嵌套运行结果: 882			
* 子程序嵌套运行结果: 884			
* 子程序嵌套运行结果: 886			
* 子程序嵌套运行结果: 888			
* 子程序嵌套运行结果: 890			
* 子程序嵌套运行结果: 892			

图 14-9 “子程序递归调用”运行结果

14.3 子程序的参数

14.3.1 子程序的参数建立与返回值

一个子程序总是要被其他程序调用的，这就存在一个其他程序与子程序数据互动的问题，

其他程序可能要将一些数据传递到子程序中供子程序使用,子程序对处理的结果有可能也要反馈到调用它的程序中,这就引入了子程序的参数和返回值的概念,子程序的参数用来保证其他程序的数据能够传递到子程序中,返回值则保证将子程序的处理结果返回到其他程序中。

◎子程序的参数在当前子程序内可以当作变量引用;其返回值可以赋值给调用它的程序中的一个变量

◎无论参数还是返回值,在使用的时候一定注意数据类型必须与定义的一致。

由于事件子程序的参数个数、参数类型及返回值是系统定义的,不允许用户修改,所以下面介绍的是用户自定义子程序的参数和返回值的使用。

1. 子程序的参数建立

刚建立的新子程序是没有参数的,如果要为子程序添加参数,在子程序名称上按回车键就可以增加一个空白参数定义,根据需要修改该参数的名称和数据类型即可。如果定义的参数不需要了,选中该参数行,按下“Del”删除键就可以了。

还有一种方法就是使用鼠标右键菜单来建立和删除参数。如图 14 - 10 所示。

子程序名	返回值类型	公开	备注		
加法子程序					
参数名	类型	参考	可空	数组	备注
N. 新子程序 Ctrl+N					
U. 撤消 Ctrl+Z					
C. 复制 Ctrl+C					
T. 剪切 Ctrl+X					
P. 粘贴 Ctrl+V					
E. 删除行 F10					
I. 插入新行 Ins					

图 14 - 10 鼠标右键菜单

右键点击程序名或者点击某个参数,在弹出的菜单中选择“插入新行”选项,即可加入一个新参数;右键点击某个参数,在弹出的菜单中选择“删除行”选项,即可删除选中的参数。

从图 14 - 10 中可以看出,第三种比较简便的方法是,点击程序名或者点击某个参数,按下“Insert”键即可加入一个新参数;点击某个参数,按下“F10”键即可删除选中的参数。

子程序参数类型可以根据程序需要自行设定,和正常的变量的类型设定是一样的。

◎参数的“类型”如果为空,系统默认为整数型参数。

◎子程序如需要接收参数数据,必须先在于程序定义表中参数表部分定义与欲接收数据数目相同的参数。调用子程序时所传递过来的数据将被顺序地填入对应的参数中。

◎如果传递过来的数据与子程序对应位置处的参数数据类型不一致,在可以互相转换时,系统将自动进行转换,否则会产生运行时错误。

◎子程序定义的各参数的数据类型及参数数目决定了该子程序所能够接收的参数数据的类型和数目,具有参数的子程序在被调用时必须提供与参数数目相同的数据。

◎参数仅能在子程序内部使用,使用方法等同于变量。

2. 子程序的返回值

子程序可以返回数据,但必须首先定义返回数据的类型,并且在子程序中使用“返回()”命令来将处理后的结果返回给调用它的程序使用。

每次调用子程序时,子程序中的所有语句都将被从第一条开始顺序执行,当执行到子程序尾部或者遇到“返回”命令时即返回到调用此子程序语句的下一条语句处。

子程序的返回值缺省定义为空,表示该子程序没有返回值给调用它的程序,这样子程序执行的尾部或者执行到返回命令“返回()”,就退出该子程序,返回到调用它的程序中。

如果要为子程序定义返回值,需要在子程序的“返回值类型”单元格中定义其返回值的数据类型,则必须使用返回命令退出子程序,并且在子程序任意一个结束分支中,都必须使用返回命令将欲返回的值回传。

◎返回命令中的参数数据类型必须和子程序的返回值类型一致。

在图 14-11 所示例子的子程序定义中,子程序名为“子程序_加法”,返回值类型为“小数型”,子程序设定了两个小数型参数,分别为“加数”和“被加数”。

子程序名	返回值类型	公开	备注		
子程序_加法	小数型				
参数名	类型	参考	可空	数组	备注
加数	小数型				
被加数	小数型				
返回 (加数 + 被加数)					

图 14-11 子程序的返回值及参数设定

该子程序使用“返回(加数 + 被加数)”命令,即对参数做了相加运算,又将运算结果回传。

对于不带参数的子程序调用在 14.2 节中已经介绍过,不再介绍,下面介绍一下带参数的子程序调用的方法。

在图 14-12 所示例子中,子程序“子程序_加法”既有参数又有返回值,在调用子程序的时候必须给该子程序提供参数值,且数据类型一致,如“子程序_加法 (200.5, 312.666)”,当然这里提供参数值的时候也可以以变量的方式提供。

子程序名	返回值类型	公开	备 注		
_按钮1_被单击					
变量名	类 型	静态	数组	备 注	
和	小数型				
和 = 子程序_加法 (200.5, 312.666)					
信息框 (到文本 (和), 0, “求和”)					

子程序名	返回值类型	公开	备 注		
子程序_加法	小数型				
参数名	类 型	参考	可空	数组	备 注
加数	小数型				
被加数	小数型				
返回 (加数 + 被加数)					

图 14-12 调用带参数的子程序

子程序具有返回值属性,所以在调用它的程序中,需要以赋值的方式将返回值取出,如“和 = 子程序_加法(200.5,312.666)”。

14.3.2 子程序参数的可空属性

易语言自带的支持库命令中有一些参数可以被省略,一般查找帮助时可以看到可省略的部分用中括号“[]”括起来了。如“时间到文本()”命令的第二个参数是可以省略的,它也可

有一个默认的数据,此命令帮助如下:

调用格式:〈文本型〉时间到文本(欲转换到文本的时间,[转换部分])

将指定时间转换为文本并返回。

参数<1>的名称为“欲转换到文本的时间”,类型为“日期时间型(date)”。

参数<2>的名称为“转换部分”,类型为“整数型(int)”,可以被省略。参数值可以为以下常量:1. #全部转换、2. #日期部分、3. #时间部分。如果省略了本参数,默认为“1. #全部转换”。

用户自定义子程序也可以实现提供缺省参数这样的功能。

在设计子程序的时候,有些参数可能经常用到相同的数据。如果每次调用该子程序,均需手工指定此数据作为参数初始值,那是一件非常繁琐的事情。在这种情况下,可以将此数据作为该参数的默认值,内置在子程序中,并将参数属性设为“可空”类型。当子程序被调用时,该参数没有指定具体的初始值,就为该参数赋予默认值参与运算。如果某项参数的“可空”类型未被设置,在调用该子程序时必须为此参数提供初始值数据。

如果要将某个参数定义“可空”属性,将该参数对应的“可空”属性打上“√”即可。如果需要检测当前子程序被调用时,该参数是否传递了数据,可用“是否为空()”命令进行确认。

如果一个参数的“可空”属性为“真”,那么在调用该子程序时,可以不为此参数传递数据。主要用作支持具有默认值的参数,也可以在为子程序添加了新参数后又不想去更改以前调用此子程序的语句时使用。

下面编写一个可以进行四则运算的子程序,来了解“可空”属性的实际应用。

【范例 14-1】实现两位数的四则运算,前两个参数提供运算的值,第三个参数提供运算的规则,即是加、减、乘、除中的一个,如果第三个参数为空,则缺省为加法运算。具体参考例程 14-1。

程序通过一个单击按钮的动作来显示运算结果。下面来建立四则运算程序。

(1) 新建一个易程序,在窗口中添加 3 个编辑框组件、4 个按钮组件和 5 个标签组件,将按钮组件的标题分别改为“+”、“-”、“×”和“÷”,2 个标签组件的标题改为“运算符”和“=”,另 3 个标签组件的标题改为“数值 1”、“数值 2”和“结果”。如图 14-13 所示。

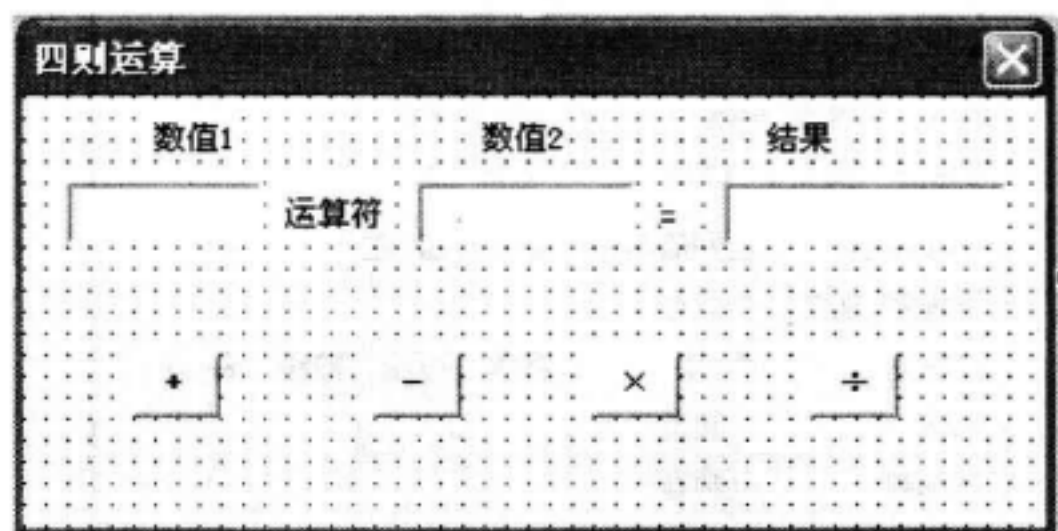


图 14-13 “四则运算”窗体设计

(2) 切换到代码编辑界面,新建一个子程序,将子程序名改为“子程序_四则运算”。给该子程序定义 3 个参数,其中 2 个小数型参数,参数名分别为“数值 1”、“数值 2”,1 个整数型参数,参数名为“运算符”,将“运算符”参数的“可空”属性被设置,在子程序中输入程序代码,如图 14-14 所示。

“子程序_四则运算”简要说明:

① 通过对“是否为空(运算符)”命令的判断来给变量“运算符”初始化。

子程序名	返回值类型	公开	备 注		
子程序_四则运算	小数型		两位数四则运算，运算符0、1、2、3分别代表+、-、*、/，缺省为0，加运算		
参数名	类 型	参 考	可 空	数 组	备 注
数值1	小数型				
数值2	小数型				
运算符	整数型		✓		
如果真 (是否为空 (运算符))					
运算符 = 0					
如果真 (运算符 ≥ 4)					
运算符 = 4					
返回 (多项选择 (运算符 + 1, 数值1 + 数值2, 数值1 - 数值2, 数值1 × 数值2, 数值1 ÷ 数值2, 3.4e+038))					

图 14-14 “四则运算”子程序代码

② 通过对“运算符≥ 4”的判断来达到程序容错的目的，提高程序的可用性。如果变量“运算符”赋值大于等于4 成立的话，就恒定为“运算符 = 4”，这个命令对四则运算本身作用不大，主要是增加程序的容错性，保证只要给“运算符”的赋值超出 3(四则运算)，就返回一个固定的值“3.4e + 038”，表示出错。

③ 在调用子程序时，对参数的赋值易语言只检查数据类型是否符合，值的大小是在程序运行中来检查的，如在本程序中给变量“运算符”传递一个大于等于 4 的值，尽管没有对应的运算符号，在编译时也不会告警，只有在程序运行的时候，命令“多项选择()运算符 + 1, 数值1 + 数值2, 数值1 - 数值2, 数值1 × 数值2, 数值1 ÷ 数值2”才会提醒出错。为解决这个问题，在“多项选择()”命令中增加一个参数来识别运算符是否符合规定，即使用命令“多项选择(运算符 + 1, 数值1 + 数值2, 数值1 - 数值2, 数值1 × 数值2, 数值1 ÷ 数值2, 3.4e + 038)”，当提供了一个非四则运算的值后，子程序返回一个固定的值(3.4e + 038 为小数型数据类型能表示的最大值，这里技巧性地以这个值表示程序出错)。

④ 使用“多项选择()”命令来实现四类运算的值的返回显得程序比较有技巧，程序执行效率较高。由于该命令第一个参数表示选择的位置，且从 1 开始，所以需要“运算符 + 1”来表示(变量“运算符”的有效赋值范围是从 0-3)。

(3) 回到“_启动窗口”双击“+”按钮，然后在“_按钮_加法_被单击”子程序中输入代码。如图 14-15 所示。

子程序名	返回值类型	公开	备 注	
_按钮_加法_被单击				

变量名	类 型	静 态	数 组	备 注
结果	小数型			
运算符	整数型			

标签_运算符.标题 = 按钮_加法.标题

结果 = 子程序_四则运算 (到数值 (编辑框_数值1.内容), 到数值 (编辑框_数值2.内容),)

编辑框_结果.内容 = 到文本 (结果)

图 14-15 “_按钮_加法_被单击”子程序代码

可以看到，在“_按钮_加法_被单击”子程序中，省略了“子程序_四则运算”子程序第 3 个参数的填写。

依次给“_按钮_减法_被单击”子程序、“_按钮_乘法_被单击”子程序和“_按钮_除法_被单

击”子程序输入代码。以“_按钮_减法_被单击”子程序为例,比较和“_按钮_加法_被单击”子程序的不同。如图 14-16 所示。

子程序名		返回值类型	公开	备注
_按钮_减法_被单击				
变量名	类型	静态	数组	备注
结果	小数型			
运算符	整数型			
标签_运算符.标题 = 按钮_减法.标题				
运算符 = 1				
结果 = 子程序_四则运算 (到数值 (编辑框_数值1.内容), 到数值 (编辑框_数值2.内容), 运算符)				
编辑框_结果.内容 = 到文本 (结果)				

图 14-16 “_按钮_减法_被单击”子程序代码

可以看到在“_按钮_减法_被单击”子程序中,增加了一个给变量“运算符”赋值的命令 (“运算符=1”)来表示要进行减法运算,并且在调用“子程序_四则运算”子程序时第3个参数也被填写为变量“运算符”。

(4) 最后运行程序查看运行后的效果。程序运行时点击“+”按钮,程序运行的“子程序_四则运算”子程序第三个参数没有填参数,因此进行的是加法计算。如图 14-17 所示。

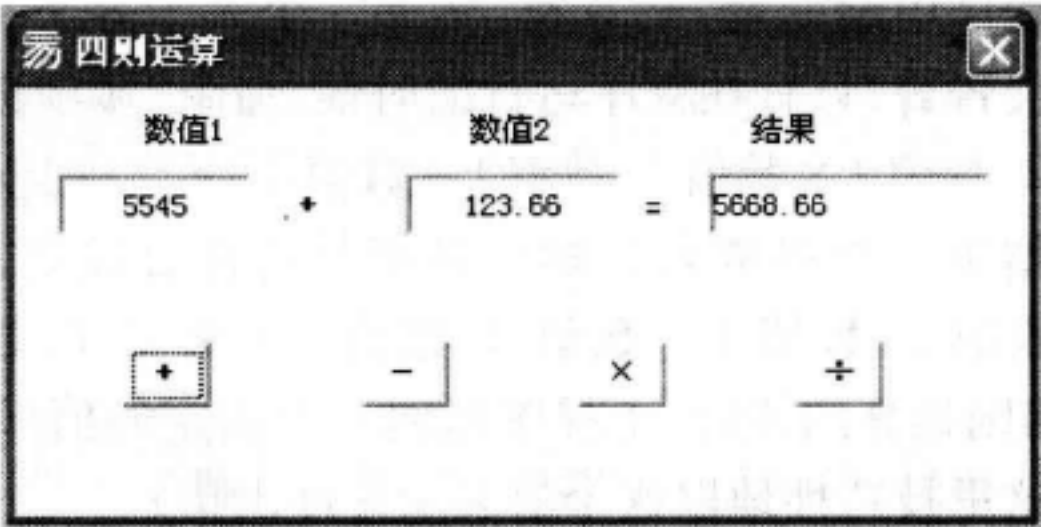


图 14-17 “加法运算”运行结果

程序运行时点击“-”按钮,程序运行的“子程序_四则运算”子程序第三个参数设置参数为1,因此进行的是减法计算。如图 14-18 所示。

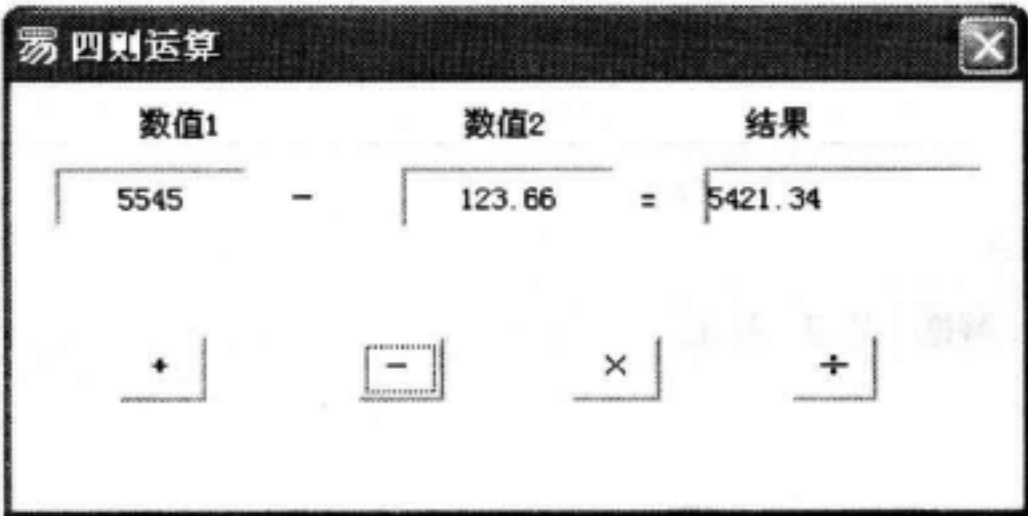


图 14-18 “减法运算”运行结果

14.3.3 子程序参数的参考属性

子程序参数的“参考”属性用于设置系统为当前子程序参数传递数据时是否为传递指向数据的指针。

通俗的说法是,在易语言子程序参数的传递中有两种方法,一种是直接将参数的值传递到

子程序中使用,那么在子程序中使用该参数并且值发生了变化,也不会影响该参数原有的值;一种是将指向该参数值的地址(即保存该参数值的位置)传递到子程序,子程序在使用该参数,且修改了值的大小时,就会改变原参数指向位置的值,即在子程序中对该参数值的修改结果能够传递到子程序外。

易语言中,如果所传递过来的参数数据为数组、用户定义数据类型、库定义数据类型、文本型、字节集型数据,则无论此属性是否为“真”(选中),都将传递指针。

◎因此在使用这些数据类型进行传递数据时,要注意是否允许在子程序中修改数据,且是否已经做了修改,否则,会因为数据被子程序做了修改而不知道,造成程序错误。这类错误在找错的时候很难发现。

如果所传递过来数据的类型与相应位置处参数的数据类型不一致但可以相互转换,则在数据被实际传递前,系统将首先自动转换,然后再进行传递。例如将“整数型”数据传递到“小数型”的参数中,则在数据被实际传递前,系统首先自动将“整数型”数据转换为“小数型”数据,然后再进行传递。

◎因此在这种情况下,即使本属性为“真”,系统也无法传递指向原数据的指针,只能传递数据本身。因此,如果想通过子程序参数的参考属性来增加程序的技巧性,或者必须使用这个属性,最好保持传递过来的数据类型和相应位置 chure 处的参数类型的一致性,避免发生不可控错误,不易查错。

如果系统将数据指针成功地传递过来,那么在子程序中对此参数内容的更改将会相应地反映到调用子程序时所提供的参数数据上。

给子程序参数定义“参考”属性,只要将属性相应位置打上“√”即可。

【范例 14-2】例程 14-2 演示子程序参数的“参考”属性的作用。

“子程序_参考属性试验”中代码很简单,就是对传递来的整数数据进行了加法运算,以便观测参数的变化情况。如图 14-19 所示。

子程序名	返回值类型	公开	备 注		
子程序_参考属性试验					
参数名	类 型	参 考	可 空	数 组	备 注
参考值	整数型	√			
非参考值	整数型				
参考值 = 参考值 + 100					
非参考值 = 非参考值 + 100					

图 14-19 “子程序_参考属性试验”代码

在定义的“_按钮_测试_被单击”子程序中,定义了两个变量“变量_参考”、“变量_非参考”用来临时保存传递前参数的值。通过比较子程序执行前后这两个参数值的变化,可以观测参数“参考”属性的作用。如图 14-20 所示。

运行程序,观察运行结果,可以看到“参考”属性对变量的影响。如图 14-21 所示。

从显示结果可以看出,对应“参考属性”的值,因子程序内部对相应参数的操作而被修改,而对应“非参考属性”的值则没有发生任何变化。因此,可以这么理解,使用子程序参数“参考”属性后,子程序中参数值的变化会影响到子程序外数据的变化。

14.3.4 子程序参数的数组属性

子程序参数也可以接收数组,实现了批量数据在子程序调用时的传递。子程序参数定义

子程序名		返回值类型	公开	备注
_按钮_测试_被单击				
变量名	类型	静态	数组	备注
变量_参考	整数型			
变量_非参考	整数型			
变量_参考 = 到数值 (编辑框_参考属性.内容)				
变量_非参考 = 到数值 (编辑框_非参考属性.内容)				
子程序_参考属性试验 (变量_参考, 变量_非参考)				
编辑框_参考属性结果.内容 = 到文本 (变量_参考)				
编辑框_非参考属性结果.内容 = 到文本 (变量_非参考)				

图 14-20 观测“参考”属性的作用



图 14-21 “参考”属性的作用运行结果

了“数组”属性后,就可以给这个参数提供一个数组变量。如果给子程序的参数提供了“数组”型变量,则该参数就自动具有了“参考”属性。

定义参数的“数组”属性,在该参数的数组属性上打“√”即可。

【范例 14-3】例程 14-3 通过实现一个编辑框的内容传递到一个列表框中,来了解“数组”参数的作用。在此程序中有一个子程序“子程序_数组属性”,来实现编辑框内容采集到数组的过程(每行数据作为列表框的一项)。如图 14-22 所示。

子程序名		返回值类型		公开	备 注	
子程序_数组属性		逻辑型				
参数名		类 型		参考	可空	数组 备 注
源数据		文本型				
返回数据		文本型			✓	
变量名	类 型	静态	数组	备 注		
值	文本型					
返回数据 = 分割文本 (源数据, #换行符,)						
返回 (真)						

图 14-22 “子程序_数组属性”代码

为提高程序可读性,新建了一个子程序“子程序_文本到列表框”,实现文本数组数据添加到列表框的功能。如图 14-23 所示。

这样在“按钮被单击”事件中的程序结构就清晰明了。如图 14-24 所示。

最后试运行程序,查看运行效果,如图 14-25 所示。

运行后,可以看到编辑框内的文本数据传递到了列表框中。可以看出,在调用子程序后,“返回数据”数组参数自动具有了参考属性,“数组”变量中成员的值发生了改变。

子程序名	返回值类型	公开	备注		
子程序_文本到列表框					
参数名	类型	参考	可空	数组	备注
源数据	文本型			✓	
目标列表框	列表框				

变量名	类型	静态	数组	备注
计次	整数型			

目标列表框.清空 0 清空列表框

通过循环依次往列表框添加数据

 计次循环首 (取数组成员数 (源数据), 计次)

 目标列表框.加入项目 (源数据 [计次],)

 计次循环尾 0

图 14-23 “子程序_文本到列表框”代码

子程序名	返回值类型		公开	备注
_按钮_调用_被单击				

变量名	类型	静态	数组	备注
数组	文本型		0	
文本	文本型			

文本 = 编辑框_源.内容

子程序_数组属性 (文本, 数组)

子程序_文本到列表框 (数组, 列表框_目的)

图 14-24 “按钮被单击”事件代码

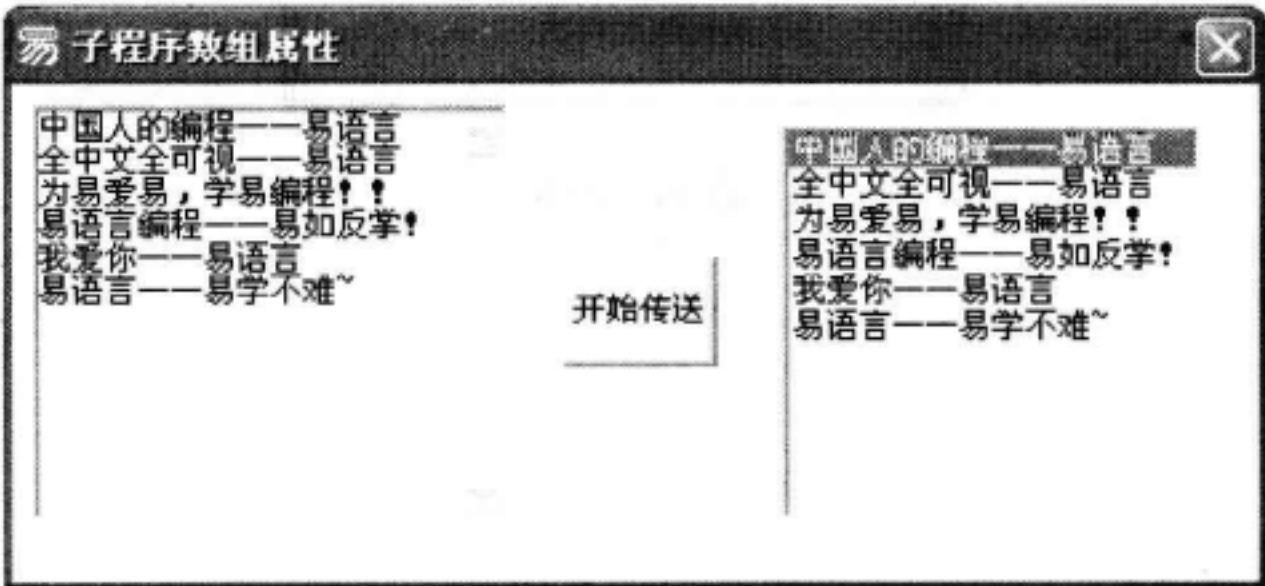


图 14-25 “数组属性”运行效果

14.4 子程序的指针应用

在易语言中,有部分命令必须以子程序在内存中的起始地址作为参数,如“启动线程()”命令;使用某些外部 API 函数和调用某些 DLL 函数的时候,也同样需要知道调用的子程序在内存中的起始地址。在易语言中,以子程序指针型变量记录子程序在内存中的起始地址,赋值方法为“&”+子程序名。具体使用方法可以参见例程 14-4,其中“启动线程”命令的使用如图 14-26 所示。

赋值之后,“指针变量”(类型为“子程序指针型”)存储的就是“子程序_指针应用”的可执行代码首地址。

命令“启动线程(子程序指针 欲执行的子程序,[整数型 参数数据],[整数型变量 线程

子程序名		返回值类型	公开	备 注	
__启动窗口_创建完毕					

变量名	类 型	静态	数组	备 注	
指针变量	子程序指针				

指针变量 = @子程序_求和
启动线程 (指针变量, 100,)

子程序名	返回值类型	公开	备 注		
子程序_求和	整数型				
参数名	类 型	参考	可空	数组	备 注
加数	整数型				

» 输出调试文本 (加数 + 300)
返回 (加数 + 300)

图 14-26 指针型变量的赋值

句柄])”是在“多线程支持库”中,需要通过菜单“工具”→“支持库配置”来添加该支持库才可使用。如图 14-27 所示。

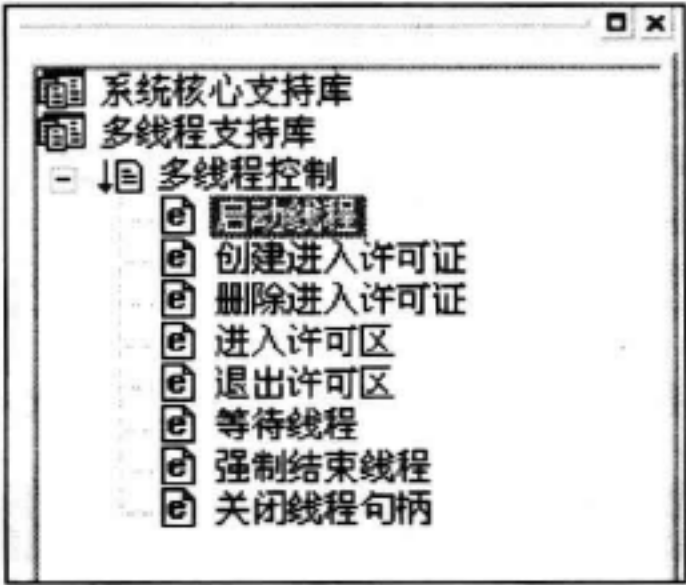


图 14-27 添加“启动线程”支持库

第 15 章 易 模 块

15.1 易模块概述

在第 13 章中,介绍了子程序的编制及其调用,知道了程序员可以将一些功能独特的、多次使用的代码构造成子程序,以便在程序的其他位置调用。但是在一个较大的项目中,可能需要多个易程序员合作开发,对于在开发中存在的一些可以共享使用的代码,他们之间如何才能实现共享调用呢?

易语言提供了这样一个在易程序中团结合作的方式,将一些具有特定功能的、需要重复使用的代码,按照一定的规范封装起来,其他人员只需要在自己的开发环境中调用即可。这种将易语言代码按照一定模式封装起来供第三方人员使用的模块化开发方式,称之为易模块。

通过使用易模块,程序开发人员可以将常用的程序代码封装为易模块,重复使用到其他程序中,或者提供给第三方用于程序开发,或者作为开发大型软件项目的一部分,最后将这些模块编译成为一个完整程序。

一个封装好的易模块,是以一个文件的形式存在的,文件扩展名为“.ec”,如“取系统时间.ec”。易模块中被封装的代码在调用时,和调用子程序的方法相同,在一个易模块中,可以只封装一个程序段的代码,也可以封装多个程序段的代码。

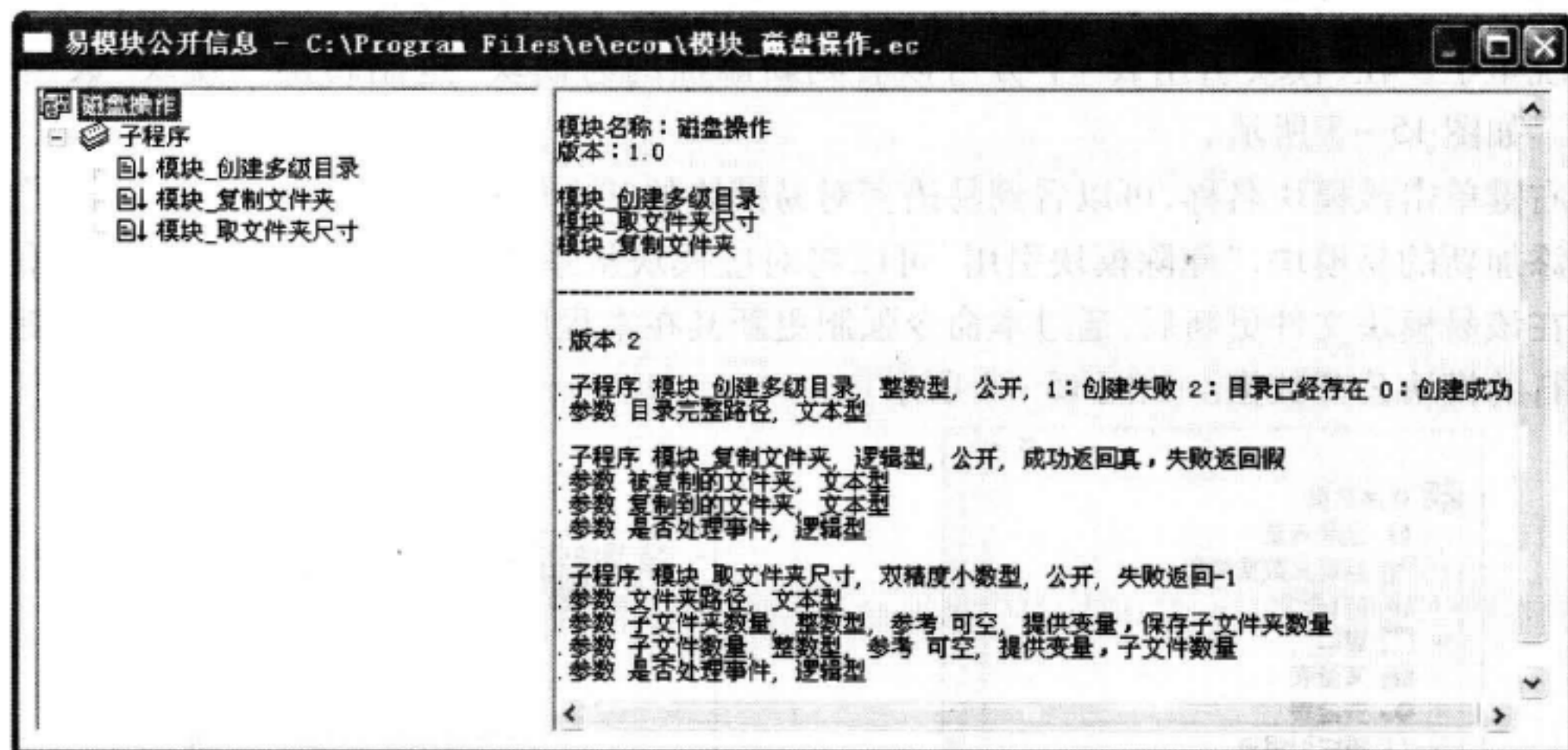


图 15-1 已封装的易模块

15.2 易模块的调用方法

对于已经编译好的易模块,要想被其他程序调用,首先要将其正确的安装到系统中。下面通过一个例子来了解易模块的安装及其调用方法。

15.2.1 易模块的安装

要安装或调用一个易模块,首先要存在这样一个易模块文件,易语言安装环境提供了一个专门存放易模块的文件夹“ecom”,完整路径为“C:\Program Files\e\ecom”。当然,易模块文件也可以保存在其他文件夹下,只是在使用的时候要注意它所在的位置,以便容易找到。

双击易语言中的“模块引用表”,或者右键单击选择“模块引用表”,在弹出的菜单中,选择“添加模块引用”项,会弹出一个选择易模块的窗口,缺省为“ecom”文件夹。如图 15-2 所示。

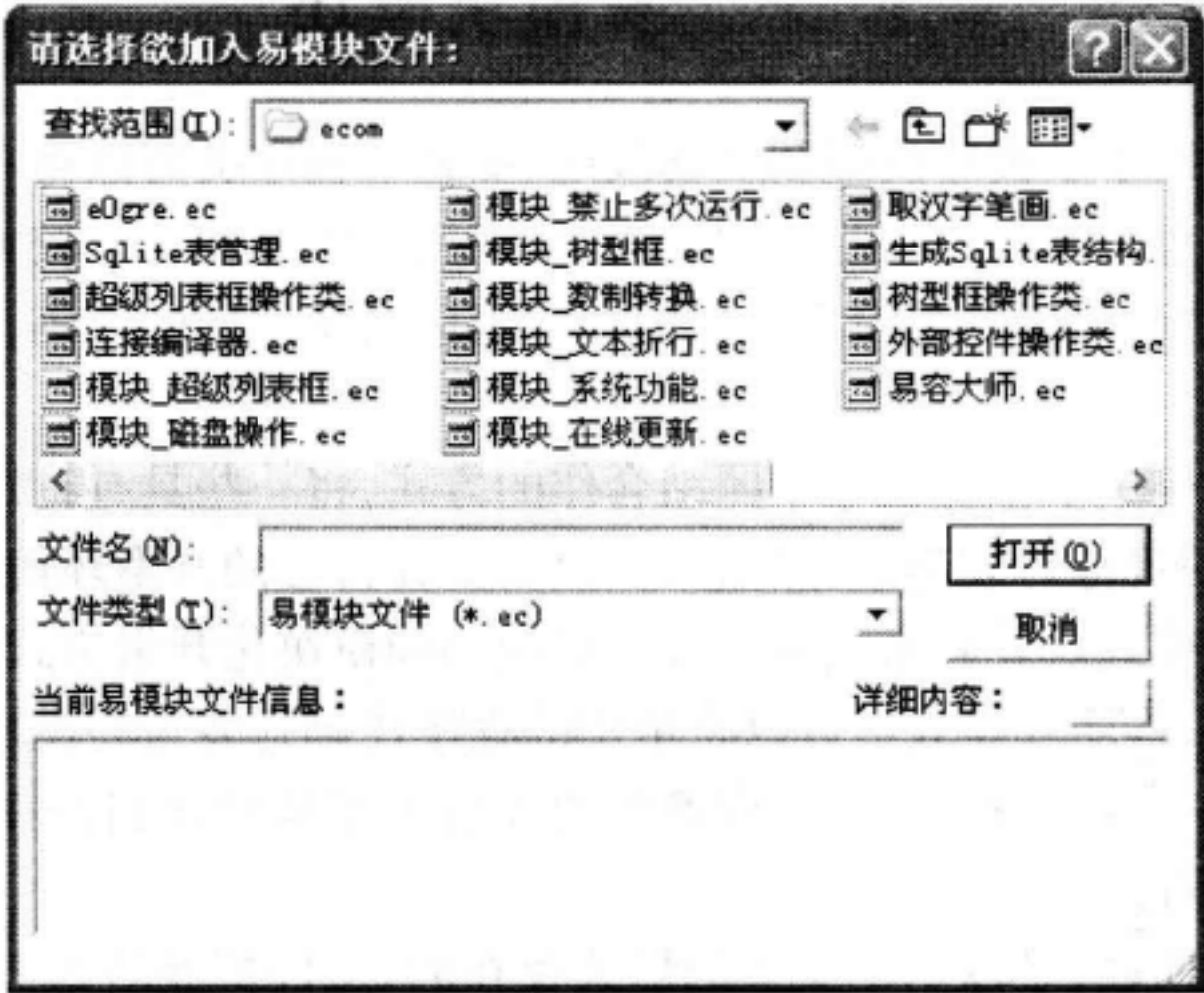


图 15-2 “选择易模块”窗口

选中“模块_数制转换.ec”,并单击“打开”按钮,就可以将易模块“模块_数制转换”添加到系统里了。在“模块引用表”下方可以看到新添加的易模块“进制转换 - 模块_数制转换.ec”。如图 15-3 所示。

右键单击该模块名称,可以看到易语言对易模块的几个管理功能。“添加模块引用”能够继续添加新的易模块,“删除模块引用”可以将对应模块从系统中删除,“更新模块引用信息”可以在该易模块文件更新后,通过本命令强制更新其在本程序中的记录信息,以便正确使用其最现行的模块公开数据。如图 15-4 所示。



图 15-3 已添加的易模块

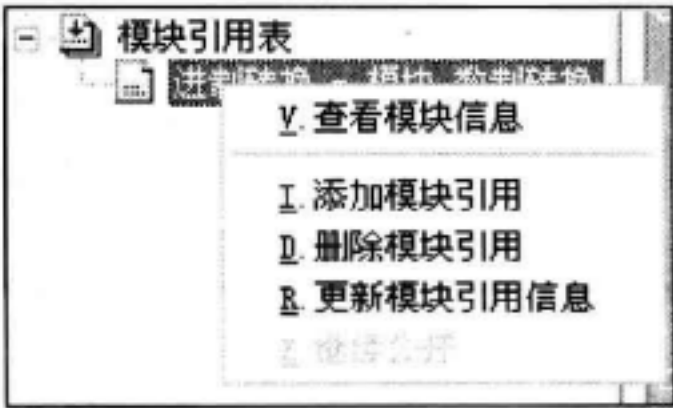


图 15-4 对易模块的管理功能

选中“查看模块信息”项,可以在“易模块公开信息”窗口中了解该模块包含的功能及其调用参数,在本例中,可以看到该模块包含一个子程序“模块_进制转换”,输入参数有三个:“被转换的文本”、“被转换进制”和“转换的进制”,输出参数为文本型。如图 15-5 所示。

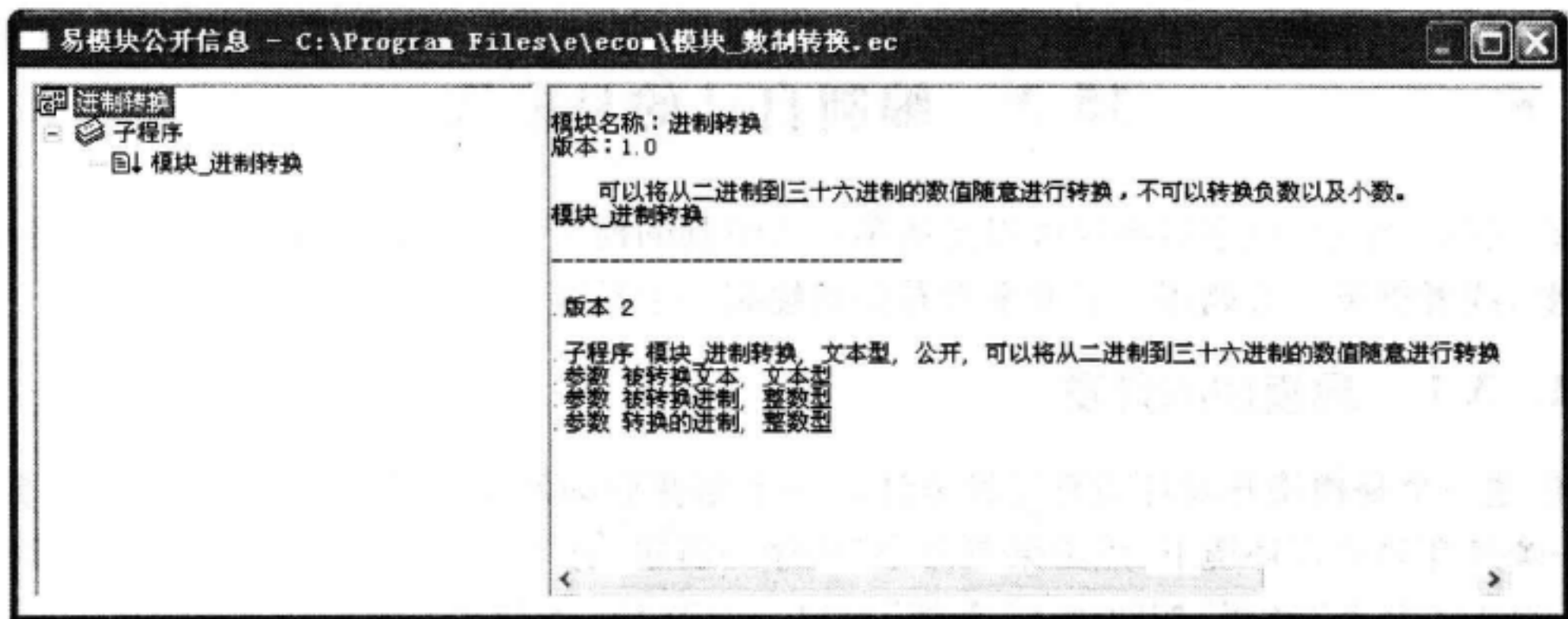


图 15-5 查看易模块信息

易模块安装完成后, 就可以调用其中的子程序了, 在编译易程序的时候, 所有被用到的易模块会一同被编译到系统中。

15.2.2 易模块的调用

易模块的调用很简单, 和使用易命令、易子程序是一样的, 只要设置好相应的参数即可。

【范例 15-1】例程 15-1 演示了一个将二进制到三十六进制的数值随意进行转换的过程, 进制转换的子程序代码如图 15-6 所示, 可以看到模块的调用“转换的文本 = 模块_进制转换(被转换的文本, 被转换的进制, 转换的进制)”和子程序的调用没有什么区别。

子程序名	返回值类型	公开	备注
按钮_进制转换_被单击			

变量名	类型	静态	数组	备注
被转换的文本	文本型			
被转换的进制	整数型			
转换的进制	整数型			
转换的文本	文本型			

被转换的文本 = 编辑框_被转换的文本.内容
被转换的进制 = 到整数 (编辑框_被转换的进制.内容)
转换的进制 = 到整数 (编辑框_转换的进制.内容)
转换的文本 = 模块_进制转换 (被转换的文本, 被转换的进制, 转换的进制)
编辑框_转换的文本.内容 = 转换的文本

图 15-6 “进制转换”代码

运行结果如图 15-7 所示, 将一个十六进制数据“16BH”转换为十进制数据“363”。

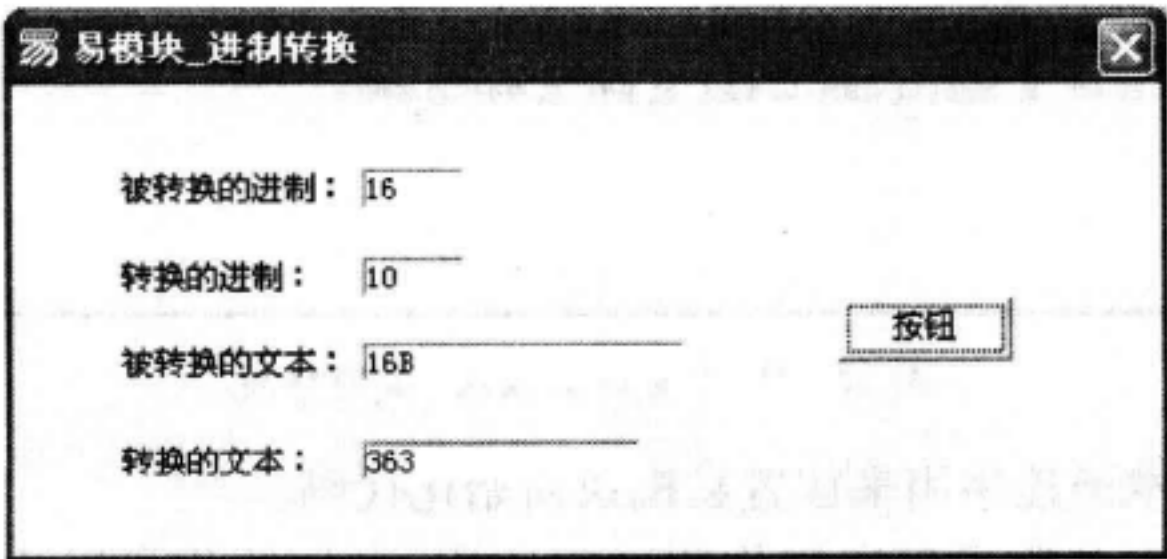


图 15-7 “进制转换”运行结果

15.3 编制自己的易模块

在实际工作中,程序员不仅可以使⽤第三⽅编制的模块,也可以⾃⾏创建易模块供⾃⾝编程需要,或者供第三⽅调⽤。下⾯来看看如何编制⼀个易模块。

15.3.1 易模块的开发

创建⼀个易模块开发环境有三种⽅法。⼀个是在启动易语⾔环境的时候进⼊“新建”窗⼝,⼀个是在易语⾔环境下,点击菜单命令“程序→新建”进⼊“新建”窗⼝,⼀个是在易语⾔环境下,点击⼯具条“新建”按钮进⼊“新建”窗⼝。如图 15-8 所示。

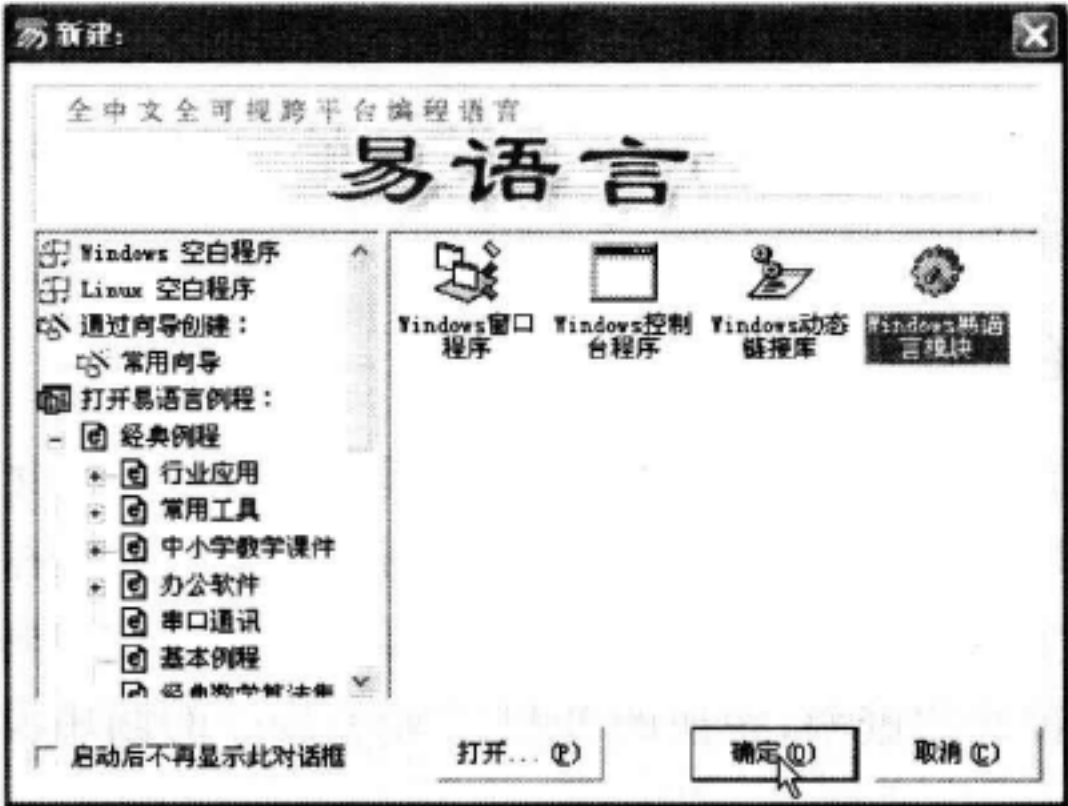


图 15-8 “新建”窗⼝

在“新建”窗⼝中,选择“Windows 易语⾔模块”并单击“确定”按钮,则进⼊易语⾔“易语⾔模块”编写环境。如图 15-9 所示。

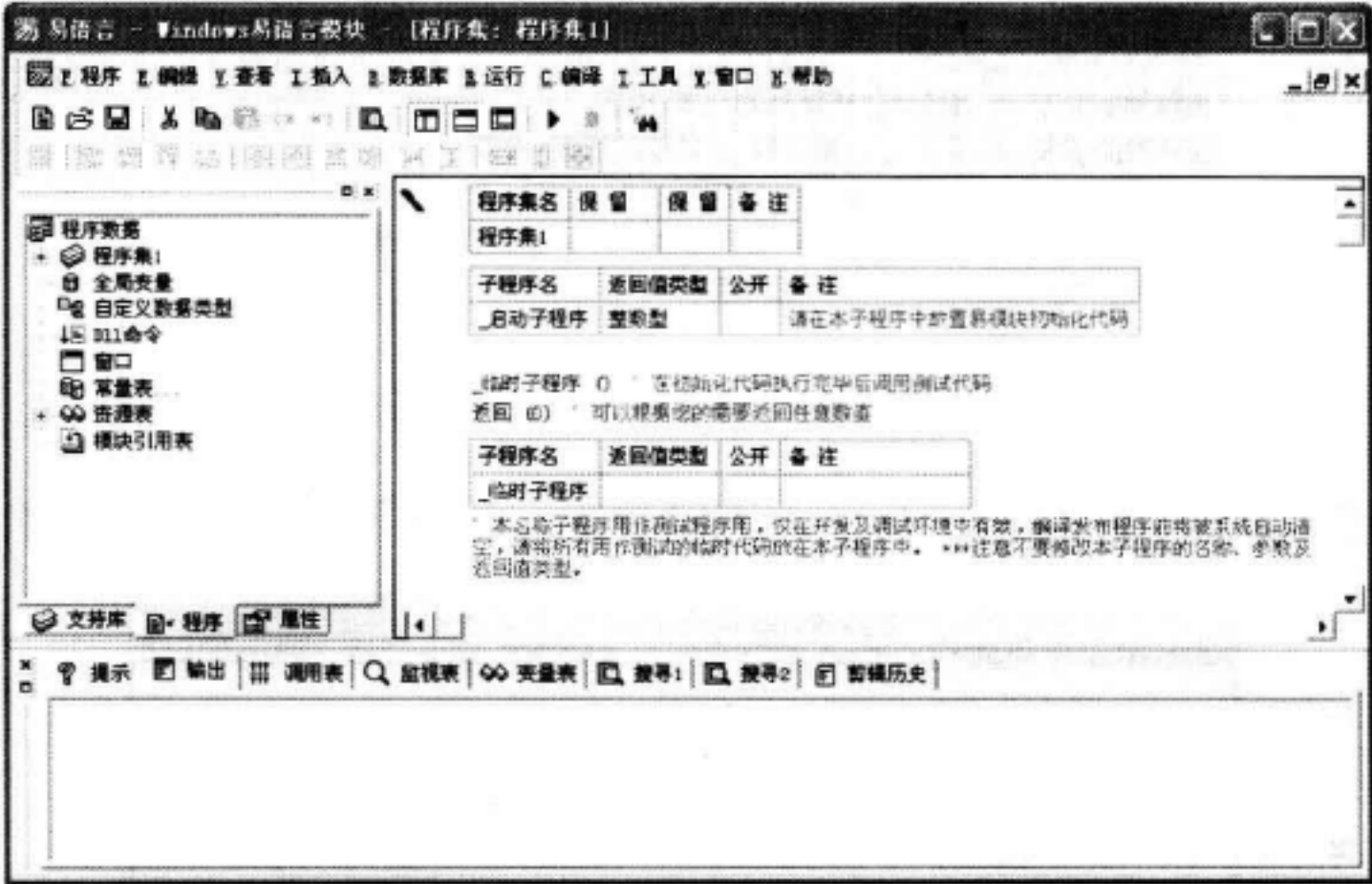


图 15-9 “易语⾔模块”编写环境

“_启动子程序”:本⼦程序⽤来放置易模块初始化代码。

“_临时⼦程序”:本名称⼦程序⽤作测试程序⽤,在初始化代码执⾏完毕后调⽤测试代码,仅在开发及调试环境中有效,编译发布程序前将被系统⾃动清空,请将所有⽤作测试的临

时代码放在本子程序中。

【注意】不要修改“_临时子程序”的名称、参数及返回值类型。

在该程序集“程序集 1”中新建一个子程序,改名为“取时间星期数”,并将子程序“公开”属性打上勾,然后输入子程序代码即完成一个独立的程序模块。如图 15-10 所示。

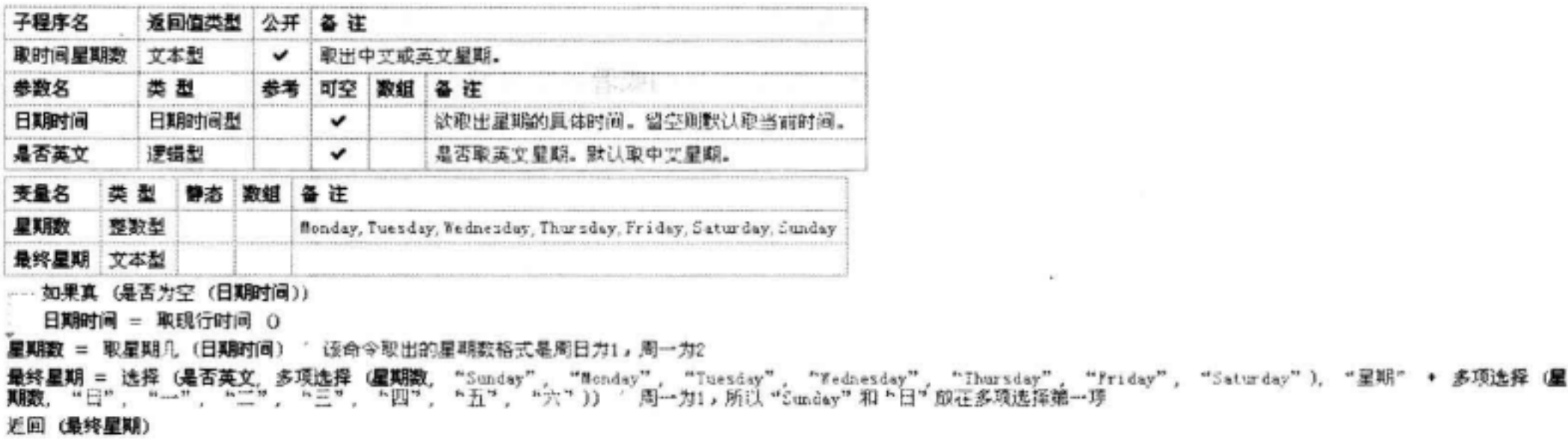


图 15-10 “独立的易模块”代码

【注意】一定要将“公开”属性打上勾,以显示这个子程序是公开的接口程序,这样其他程序才能够调用。

在“_临时子程序”中,输入调试代码“输出调试文本(取时间星期数())”,点击“运行”按钮来观测程序的执行情况。

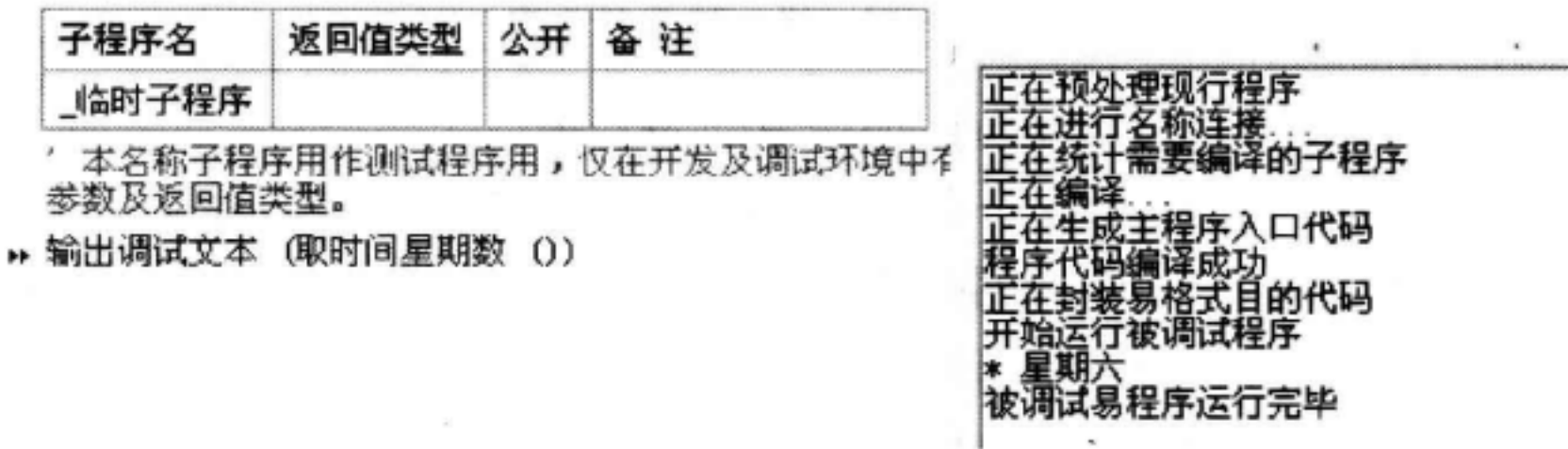


图 15-11 观测程序的执行情况

这样就完成了一个易模块代码的建立,参见例程 15-2。当然,可以继续建立其他的模块子程序,作为模块接口,供其他程序调用,只需要在子程序的“公开”属性打上勾即可。

最后,所有的代码在编译后会被封装在模块中。模块的使用者可以直接使用模块中的子程序,就如同这些子程序是自己所编写的一样。在编译易程序时所有被使用的易模块会自动被一起编译进去。不过,模块的使用者只能看到子程序的结构,无法看到子程序的完整代码,所以只能调用,而不能修改程序代码。

15.3.2 易模块的编译

易语言模块创建完成后,要想被其他程序调用,必须经过编译封装。下面来看看如何编译易模块。

首先来配置一下程序,这个过程主要是为易模块提供一下基本信息,如程序版本号、作者信息等,以便调用者参考。如果不进行配置,不影响程序的运行,但会影响使用者对程序的了解。

在“程序”菜单中选择“配置”选项,在弹出的“程序配置对话框”中填写相应程序配置的内容。如图 15-12 所示。

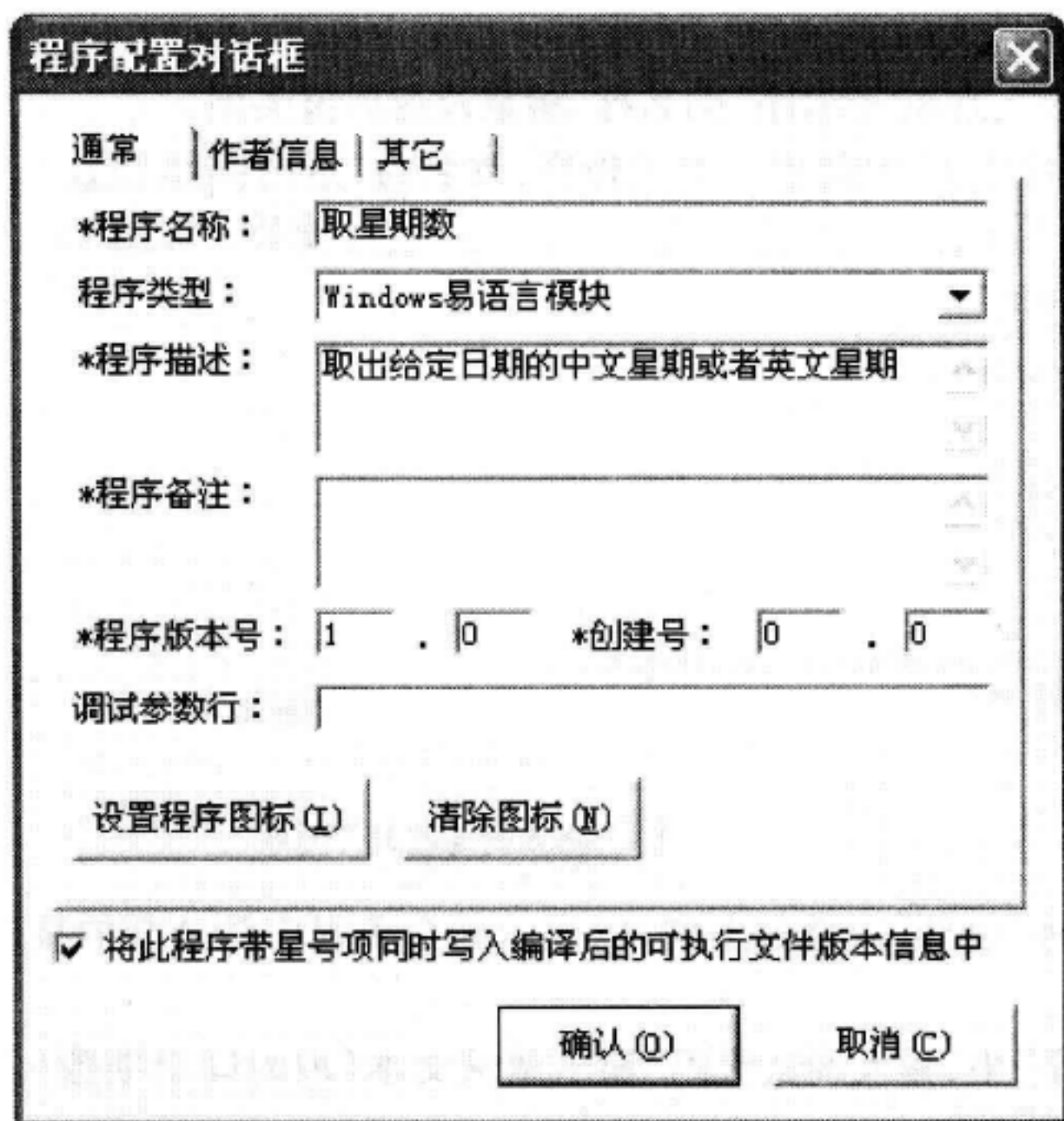


图 15-12 程序配置对话框

在“通常”选项夹中,可以对程序的一些参数进行设置。

在“程序名称”栏给这个模块起个名字,注意这个名字只是在模块中起标识作用,与调用无关,和易模块编译后保存时起得文件名不同,易模块文件名在加载时是要用到的。

在“程序类型”栏的下拉列表中选择程序类型,这里应该选择“Windows 易语言模块”。

在“程序描述”栏可以加入对这个易模块功能的简单描述。

在“程序备注”栏中可以加入对程序编写代码、功能更新的一些说明。

“程序版本号”可以保持易模块编写及升级的延续性,便于代码维护。

“调试参数行”用来设置程序调试时提供的一些参数,一般不需要设置。

“设置程序图标”按钮可以给程序加入特色图标,程序运行时显示。

“清除图标”按钮可以清除设置的特色图标。

在“作者信息”选项夹中可以填写一些与程序编制者有关的信息,如作者名称、邮政编码、联系地址、电话、传真、电子信箱以及主页地址等,便于使用易模块的人员有问题或需求时可以与编制者进行联系。如图 15-13 所示。

在“程序配置对话框”的下方,有一个复选框“将此程序带星号项同时写入编译后的可执行文件版本信息中”,可以确定是否要将配置对话框中的一些重要信息写入编译后的文件中一并发布。

配置好易模块的基本信息后,就可以对模块进行编译发布了。

点击菜单“编译→编译”命令,或者使用快捷键“F7”,都可以弹出要求给易模块命名的对话框,选择要保存的文件夹,给易模块取个能体现模块特点的名字,点击“保存”就可以了。如图 15-14 所示。

【注意】易模块缺省的文件扩展名为“.ec”。



图 15-13 “作者信息”选项卡

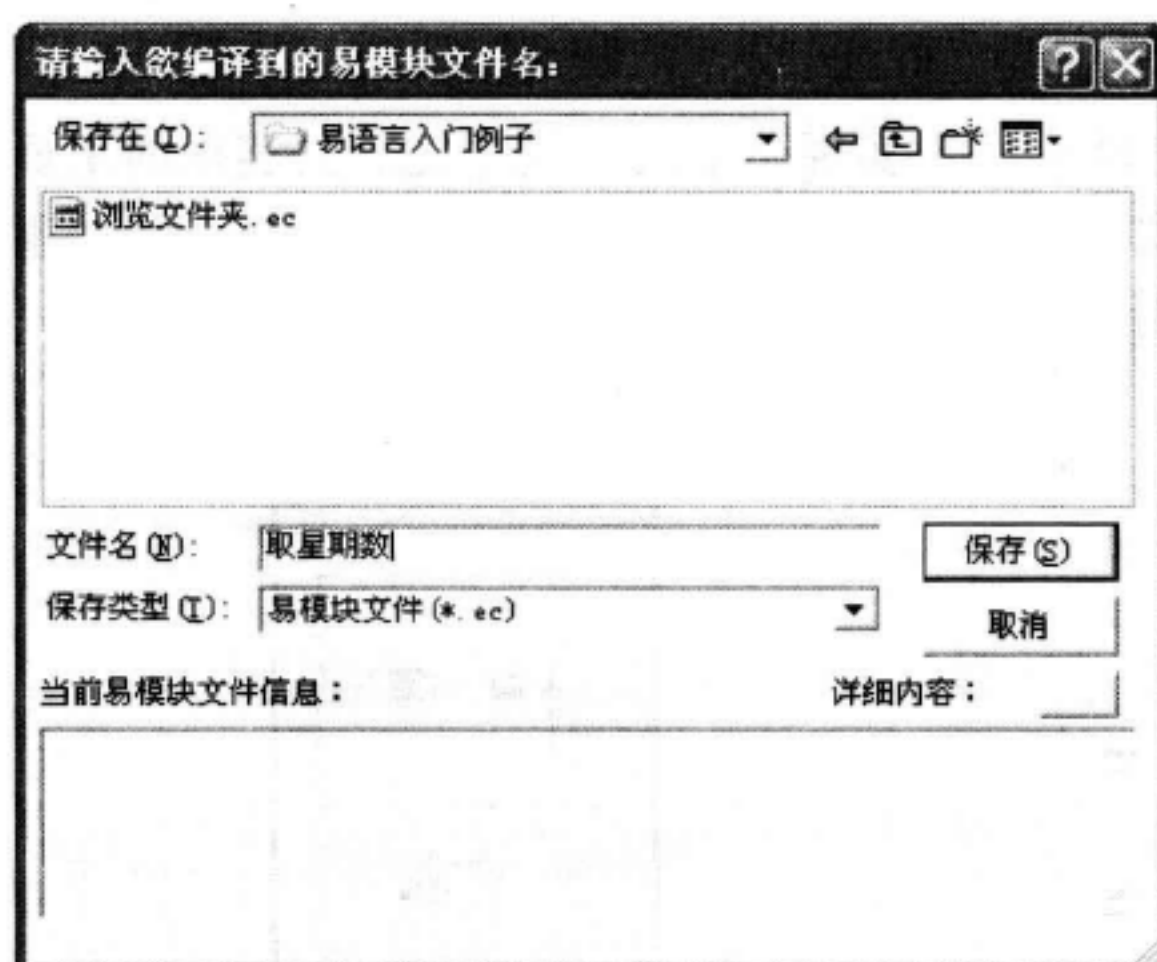


图 15-14 易模块命名及保存

点击“保存”按钮,开始对模块进行编译,就可以生成一个易模块文件“取星期数.ec”。由于易语言提供了一个缺省存放易模块的文件夹“C:\Program Files\e\ecom”,所以会提示是否要将编译完成的模块存到该文件夹下,以便以后调用。如图 15-15 所示。

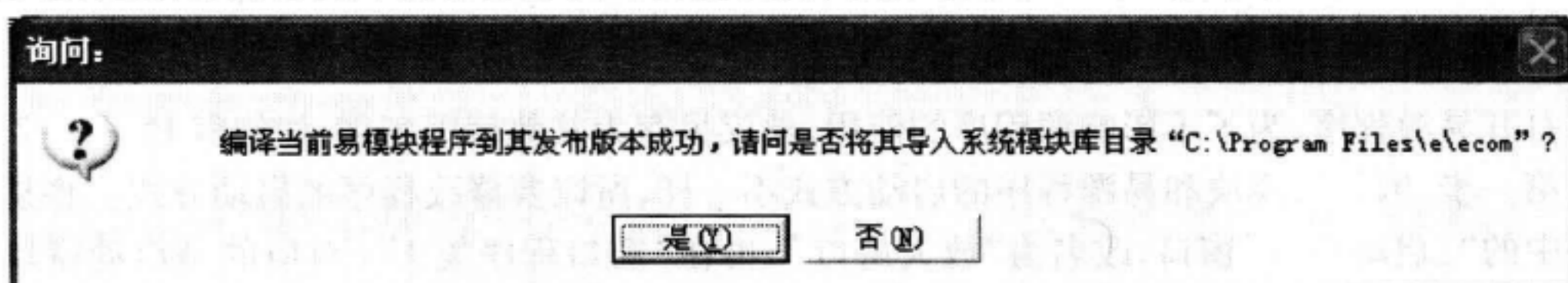


图 15-15 保存易模块至默认文件夹

参考 14.2.1 的内容,可以将刚刚编译后的易模块调入程序中查看。可以看到在“程序配置对话框”中填写的一些信息,在模块中显示出来。如图 15-16 所示。

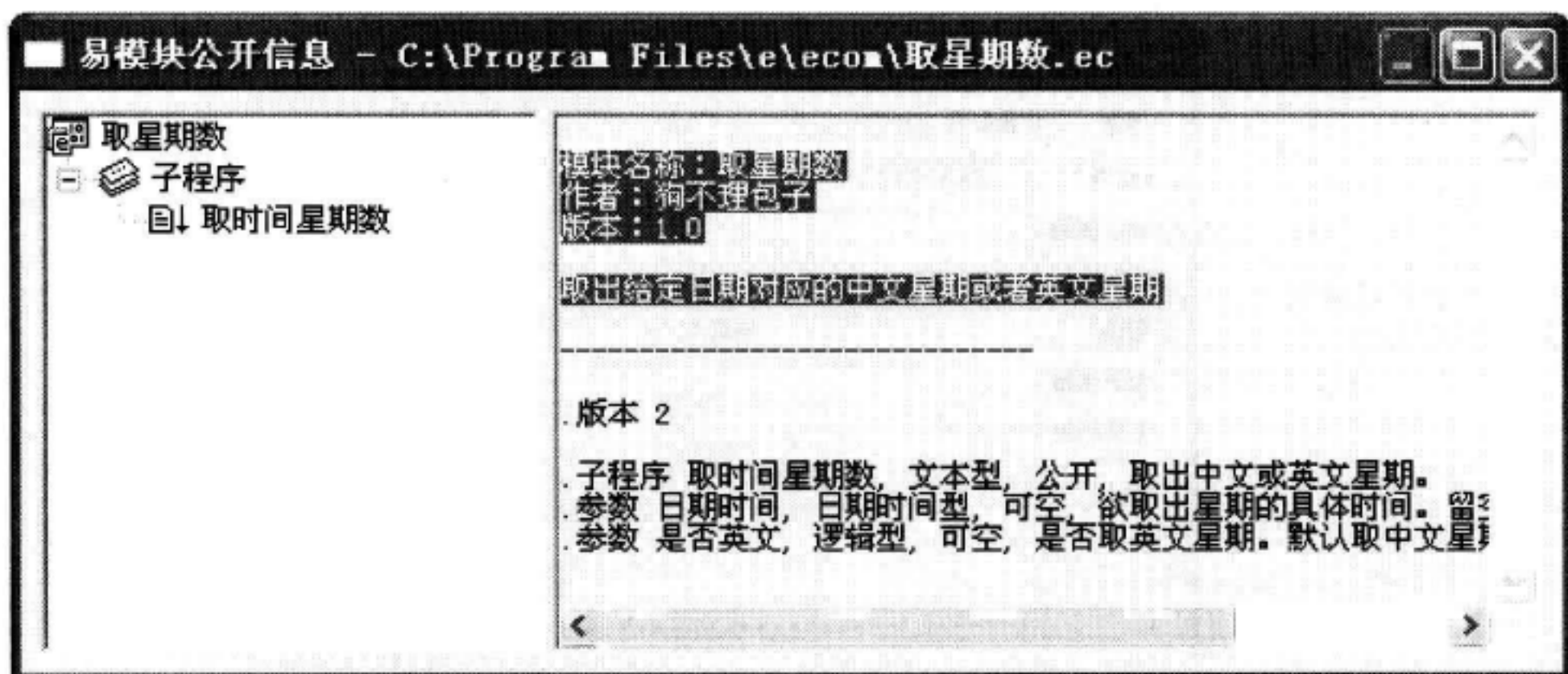


图 15-16 查看易模块的信息

15.3.3 子程序改造为易模块

前面介绍了自己如何编写易模块和如何调用易模块,下面来看看如何将一个已经编制完成的易源程序改造为易模块。

【范例 15-2】例程 15-3-1 实现一个放大镜功能。其实现的主要原理是不断通过两个时钟采集鼠标的位置,如果鼠标位置有变化,就将鼠标当前位置 50 像素见方的数据放大显示出来。程序执行效果如图 15-17 所示。

学习易语言,实现中文编程,快速提高编程能力。



图 15-17 “放大镜功能”运行效果

实现放大功能的程序源代码主要是一个时钟子程序,实现放大的原理为将一个窗口移到鼠标位置,用“快照”命令将窗口内的内容放大取出,如图 15-18 所示。

现在来将这个源程序改为一个易模块,这里主要是演示其改造方法,实现的功能比较简单,感兴趣的话,可以考虑一下提高性能的方法。

打开易源程序,为了不影响源程序的使用,建议另存为其他程序名称,如例程 15-3-2。

第一步,因为易模块和易源程序的启动方式不一样,所以要修改程序的启动方式。将易源程序中的“_启动窗口”窗口,改名为“放大窗口”,再在“窗口程序集 1”(对应的是启动窗口代码)中加入一个子程序,改名为“_启动子程序”,如图 15-19 所示。

这样,程序以后的启动就由“_启动子程序”开始了,符合了易模块的启动方式。

第二步,修改程序代码,建立易模块公开调用子程序。建立两个子程序,分别命名为“开始放大”和“结束放大”,并将“公开”属性勾选,以后在易模块中就可以调用这两个子程序了。

子程序名	返回值类型	公开	备 注
_时钟1_周期事件			
x1 = 取鼠标水平位置 0 y1 = 取鼠标垂直位置 0 长度 = 102 - 横向滚动条1.位置 窗口1.移动 (x1 - 长度 ÷ 2, y1 - 长度 ÷ 2, 长度, 长度) 如果真 (取绝对值 (x1 - x2) ≥ 1 或 取绝对值 (y1 - y2) ≥ 1) 图片框1.图片 = 快照 (窗口1.取窗口句柄 0, 图片框1.宽度, 图片框1.高度)			
子程序名	返回值类型	公开	备 注
_时钟2_周期事件			
x2 = 取鼠标水平位置 0 y2 = 取鼠标垂直位置 0			

图 15-18 “放大镜功能”代码

子程序名	返回值类型	公开	备 注
_启动子程序	整数型		请在本子程序中放置易模块初始化代码
返回 (0) 可以根据您的需要返回任意数值			

图 15-19 修改程序的启动方式

“开始放大”子程序用来载入“放大窗口”,并在成功载入的情况下,启动抓取鼠标位置的“时钟 1”和“时钟 2”,如图 15-20 所示。

子程序名	返回值类型	公开	备 注
开始放大		✓	
窗口是否载入 = 载入 (放大窗口, , 假) 记录窗口是否建立,真,建立 如果真 (窗口是否载入) 时钟2.时钟周期 = 10 时钟1.时钟周期 = 30			

图 15-20 “开始放大”子程序代码

“结束放大”子程序实现在成功载入“放大窗口”的情况下,关闭抓取鼠标位置的“时钟 1”和“时钟 2”,并将“放大窗口”销毁,如图 15-21 所示。

子程序名	返回值类型	公开	备 注
结束放大		✓	
如果真 (窗口是否载入) 窗口已经存在,可以销毁退出 窗口是否载入 = 假 时钟2.时钟周期 = 0 时钟1.时钟周期 = 0 放大窗口.销毁 0			

图 15-21 “结束放大”子程序代码

第三步,修改程序配置中的程序类型。源程序“配置”中程序类型为“Windows 窗口程序”,将其改为“Windows 易语言模块”,并将“程序名称”命名为“放大镜”即可,如图 15-22 所示。

至此,这个易模块的改写工作就完成了,编译保存为“放大镜.ec”文件就可以直接调用了。

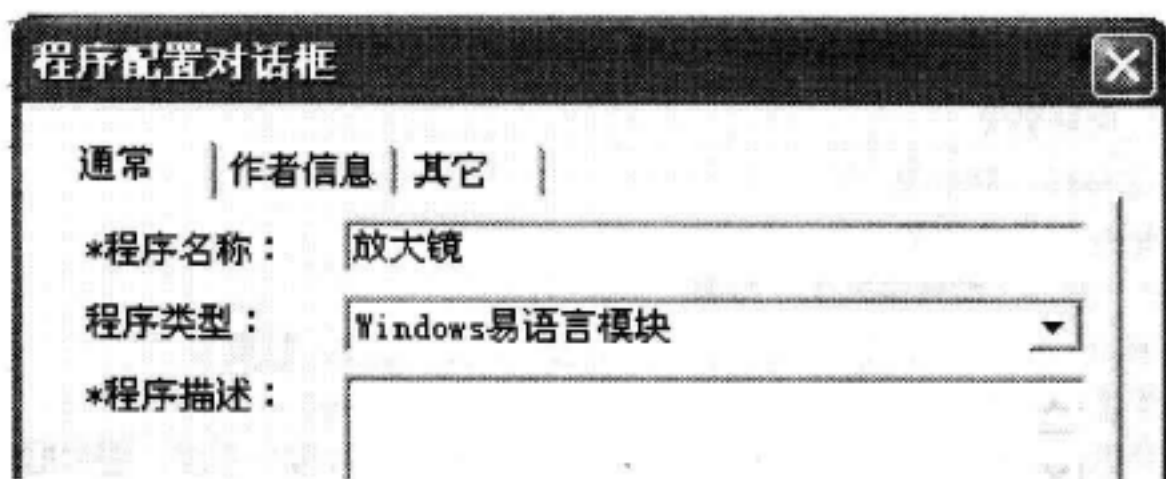


图 15-22 修改程序类型

【范例 15-3】例程 15-3-3 演示如何调用这个改造后的易模块。

建立一个易窗口程序，设置两个按钮，一个是“开始放大”按钮，一个是“结束放大”按钮，并输入代码，如图 15-23 所示。

子程序名	返回值类型	公开	备注
按钮_开始放大_被单击			
开始放大 0			
子程序名	返回值类型	公开	备注
按钮_结束放大_被单击			
结束放大 0			
子程序名	返回值类型	公开	备注
启动窗口_将被销毁			
结束放大 0			

图 15-23 “调用放大镜易模块”代码

可以看到，调用很简单。“__启动窗口_将被销毁”子程序是为了加强程序的容错性，打开“放大镜”后，程序退出的时候能够自动关闭“放大镜”。调用“放大镜”易模块的执行效果如图 15-24 所示。

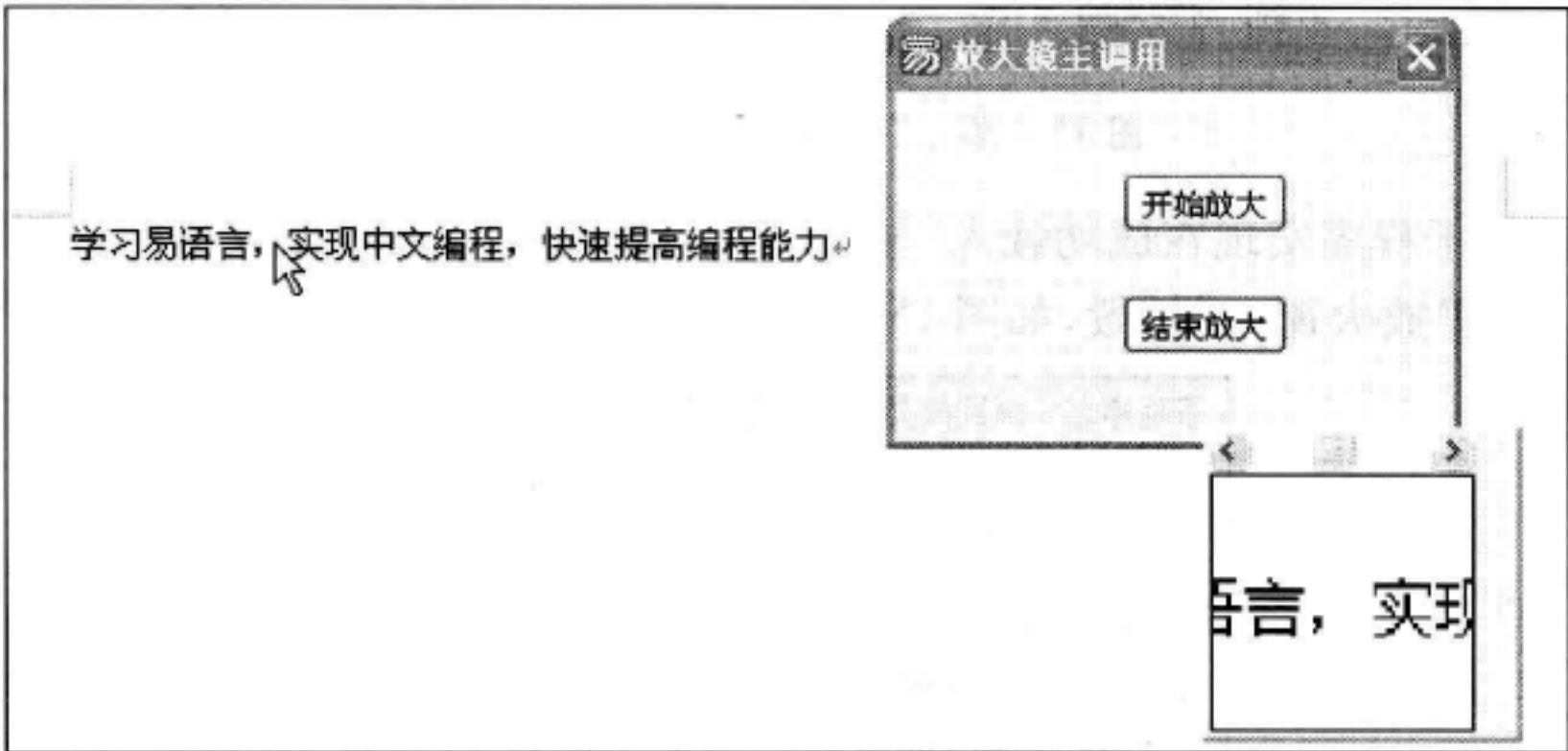


图 15-24 “调用放大镜易模块”运行结果

【注意】改写易模块时，一定要注意查看有无使用到其他程序集变量和全局变量，若有这些变量，一定要作出调整，否则会出现“变量未定义”等错误，影响模块使用。

改写易模块时，必须添加“__启动子程序”，否则不能编译。

改写易模块时，必须将程序类型修改为“Windows 易语言模块”，并为程序命名，否则不能编译。

第 16 章 API 函数调用

API 函数,也称 DLL 命令,是 Windows 系统外部动态连接库(即 DLL 库)中的命令。和 VB、VC 一样,易语言对 API 也有很好的支持。API 是 Windows 的基础,学会使用 API 就可以实现 Windows 绝大部分的功能。

16.1 API 函数概述

Windows 是一个多任务操作系统,可以把它看做一个很大的服务平台,通过调用这个服务平台上的各种服务,可以帮助应用程序实现开启窗口、管理设备、分配系统资源的功能。调用这些服务的对象称为应用程序,而实现调用的接口函数称为应用程序接口,简称为“API 函数”。

Win32 API 即为 Microsoft 32 位平台的应用程序编程接口(Application Programming Interface),作为 Microsoft 32 位平台(包括:Windows 9x, Windows NT 3.1/4.0/5.0, Windows CE)的应用程序编程接口,它是构筑所有 32 位 Windows 平台的基石,所有在 Win32 平台上运行的应用程序都可以调用这些函数。

为什么要用 API? 在 Windows 程序设计领域处于发展初期时,Windows 程序员可使用的编程工具唯有 API 函数。这些函数在程序员手中犹如“积木块”一样,可搭建出各种界面丰富、功能灵活的应用程序。不过,由于这些函数结构复杂,所以往往难以理解,而且容易误用。

随着软件技术的不断发展,在 Windows 平台上出现了很多优秀的可视化编程环境,程序员可以采用“所见即所得”的编程方式来开发具有精美用户界面和功能的应用程序。这些可视化编程环境操作简便、界面友好,比如:Visual C++, Delphi, Visual Basic、易语言等等。在这些工具中提供了大量的类库和各种控件,它们替代了 API 的神秘功能。事实上,这些类库和控件都是构筑在 Windows API 的基础上的,但它们使用方便,加速了 Windows 应用程序的开发,所以受到程序员的普遍采用。有了这些类库和控件,程序员们便可以把主要精力放在整体功能的设计上,而不必过于关注具体细节。不过,这也导致了非常多的程序员在类库面前“固步自封”,对下层 API 函数的强大功能一无所知。

实际上,程序员要想开发出更灵活、更实用、更具效率的应用程序,必然要涉及到直接使用 API 函数。虽然类库和控件使应用程序的开发容易得多,但它们只提供 Microsoft Windows 的一般功能,对于一些比较复杂和特殊的功能来说,单使用类库和控件是难以实现的,必须直接使用 API 函数来编写。因为 API 函数是构筑整个 Windows 框架的基石,只有充分理解和利用 API 函数,才能深入到 Windows 的内部,充分发挥各种 32 位平台的强大功能和灵活性,才能成功地扩展和突破类库、控件和可视开发环境的限制。近年来,随着 Microsoft 32 位平台的版本升级,Win32 API 函数的构成、功能与调用方式都有很大的发展变化。

Microsoft 的所有 32 位平台都支持统一的 API,包括函数、结构、消息、宏及接口。使用

Win32 API 不但可以开发出在各种平台上都能成功运行的应用程序,而且也可以充分利用每个平台特有的功能和属性。

在具体编程时,程序实现方式的差异依赖于相应平台的底层功能的不同。最显著的差异是某些函数只能在更强大的平台上实现其功能。例如,安全函数只能在 Windows NT 操作系统下使用。另外一些主要差别就是系统限制,比如值的范围约束,或函数可管理的项目个数等等。

标准 Win32 API 函数可以分为以下几类:窗口管理、窗口通用控制、Shell 特性、图形设备接口、系统服务、国际特性和网络服务,具体函数的说明可以参考随书光盘提供的“win32api”和“API 教程”两个文件。

16.2 如何定义 API 函数

易语言中,使用一个 API 函数前(也称 DLL 命令),首先要对该函数进行重定义,将其改造为易语言能够识别的“DLL 命令”,定义 DLL 命令涉及到以下主要属性:Dll 命令名、返回值类型、Dll 库文件名、Dll 命令在 Dll 库中的对应命令名、Dll 命令参数。

Dll 命令名:定义 DLL 命令在易语言中使用的命令名,可以随意定义,最好是定义一个方便识别的命令名。

返回值类型:定义 DLL 命令返回数据的类型。

Dll 库文件名:定义 DLL 命令所在 DLL 库的文件名称。按照 API 资料中填写,不可以自定义。

Dll 命令在 Dll 库中的对应命令名:定义 DLL 命令在 DLL 库中对应的命令名(API 函数名)。按照 API 资料中填写,不可以自定义。

Dll 命令参数:按照 API 资料中描述的进行设置,有几个参数,就添加几个参数,并且参数值的类型要和资料中的符合,参数名可以自定义。

如图 16-1 所示为一个建好的 DLL 命令,“取标题文本长度_”为定义的在易语言中使用的命令名,“整数型”为这个命令的返回值类型,“user32.dll”为 DLL 库的文件名称,“GetWindowTextLengthA”为 DLL 命令在 API 函数中的名称,“窗口句柄”为命令参数,数据类型为“整数型”。

Dll命令名	返回值类型	公开	备 注	
取标题文本长度_	整数型			
库文件名：				
user32.dll				
在库中对应命令名：				
GetWindowTextLengthA				
参数名	类 型	传址	数组	备 注
窗口句柄	整数型			

图 16-1 定义 DLL 命令

API 函数资料如何查找?

Windos 中的 DLL 命令有很多,网上也有许多 API 的帮助文件,列出了常用的 API 令的相关资料。使用 API 之前,可以先上网下载一个 API 的帮助文档。本书随书光盘提供的“win32api”和“API 教程”两个文件供参考使用。

下面通过一个范例来看看如何新建一个 DLL 命令。

【范例 16-1】例程 16-1 实现用一个 API 函数读取键盘指示灯的状态。

要调用一个“API 函数”,首先要知道这个函数的作用或函数名称,这样就可以在“API 教程”中找到这个函数的详细说明,如图 16-2 所示。

从图中可以看出,这个 API 函数的名称为“GetKeyState”,在“user32.dll”库中,它的功能是“针对已处理过的按键,在最近一次输入信息时,判断指定虚拟键的状态”,它的返回值是整

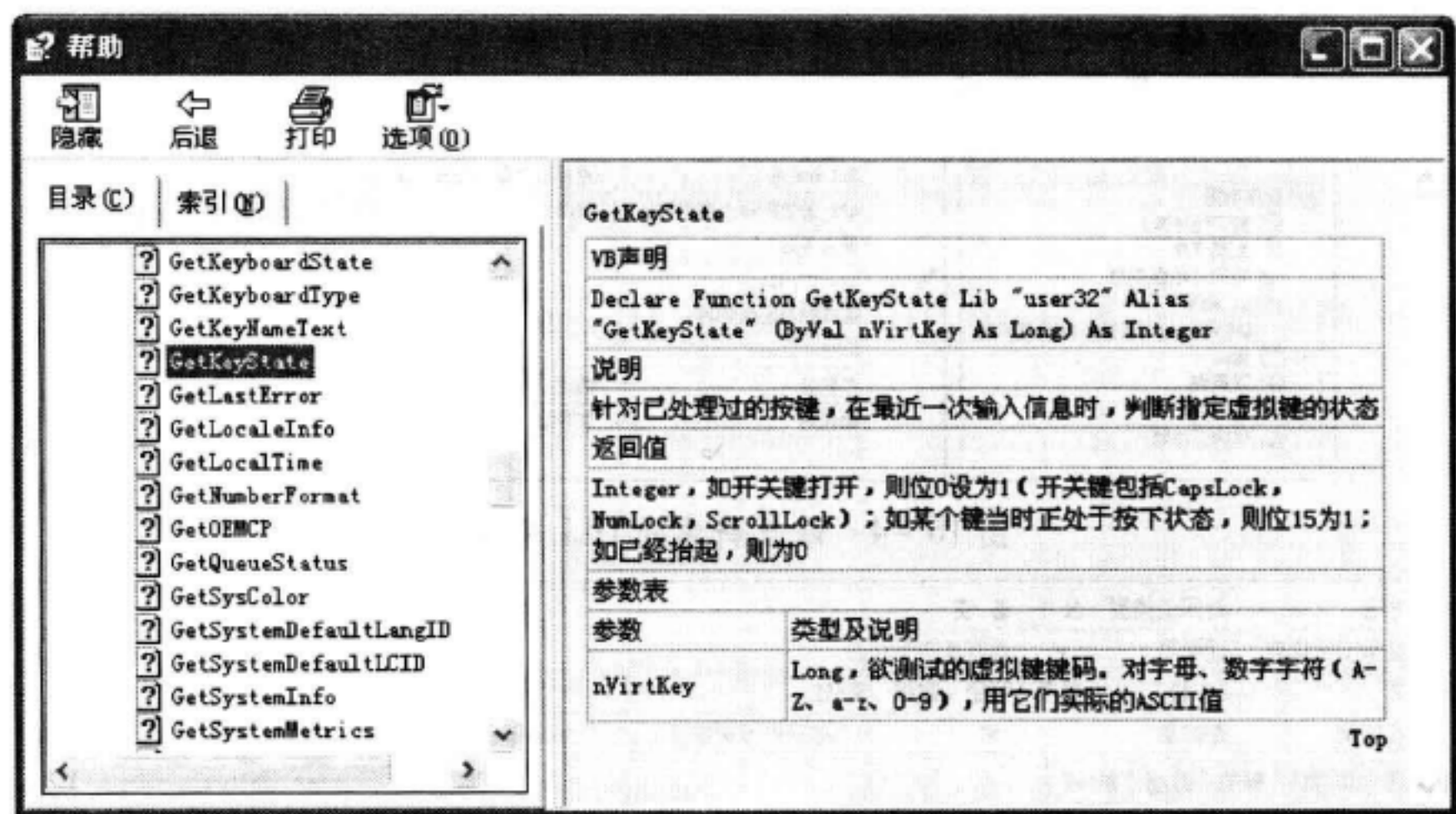


图 16-2 查看 API 函数的详细说明

型,如果开关键打开,则置位 0 为 1,参数只有一个“键代码”,存放欲测试的虚拟键代码。

了解了 API 函数的具体定义,就可以在易语言中建立一个 DLL 命令了。打开一个新的易语言窗口程序,点击菜单“插入→DLL 命令”,进入“DLL 命令定义表”,建立一个新的 DLL 命令,如图 16-3 所示。

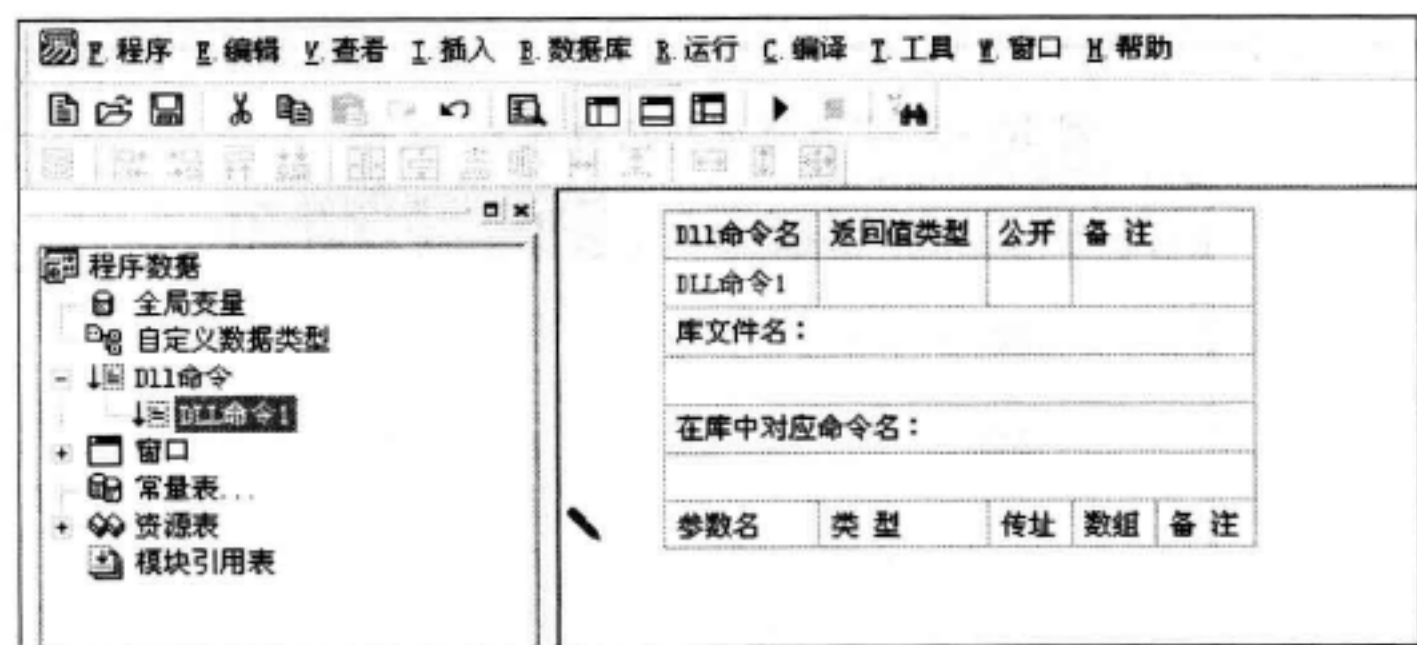


图 16-3 新建 DLL 命令

可以看到易语言 DLL 命令的基本结构。现在开始修改“DLL 命令名”为“API_最近的输入虚拟键状态”,修改“返回值类型”为“整数型”,修改“库文件名”为“user32.dll”,修改“在库中对应命令名”为“GetKeyboardState”,修改“参数名”为“键代码”,参数类型设置为“整数型”,修改后的效果如图 16-4 所示。

【注意】由于“user32.dll”函数库是 Windows 内置的标准函数库,因此在定义“库文件名”时,修改为“user32l”也可。

下面来编写取键盘指示灯状态的子程序,具体编写过程不做详细讲解,子程序代码如图 16-5 所示。

【说明】“#NumLock 键”代表“NumLock 键”的键值“144”,“#CapsLock 键”代表“CapsLock 键”的键值“20”,“#ScrollLock 键”代表“ScrollLock 键”的键值“145”(参见“易语言核心支持库常量说明”)。

执行范例 16-1 的效果如图 16-6 所示。



图 16-4 修改新建的 DLL 命令

子程序名	返回值类型	公开	备注
取键盘指示灯状态	逻辑型	✓	返回真表示开启

参数名	类型	参考	可空	数组	备注
指示灯类别	整数型		✓		0: 数字锁定键盘灯, 1: 大小写锁定键盘灯, 2: 滚动锁定键盘灯。为空默认为0

变量名	类型	静态	数组	备注
状态	整数型			

```
判断 (指示灯类别 = 0)
    小键盘灯
    状态 = API_最近的输入虚拟键状态 (#NumLock键)
判断 (指示灯类别 = 1)
    大小写灯
    状态 = API_最近的输入虚拟键状态 (#CapsLock键)
判断 (指示灯类别 = 2)
    滚动锁定灯
    状态 = API_最近的输入虚拟键状态 (#ScrollLock键)
状态=1, 开启; 状态=0, 关闭
返回 (状态 = 1)
```

图 16-5 “取键盘指示灯状态”代码

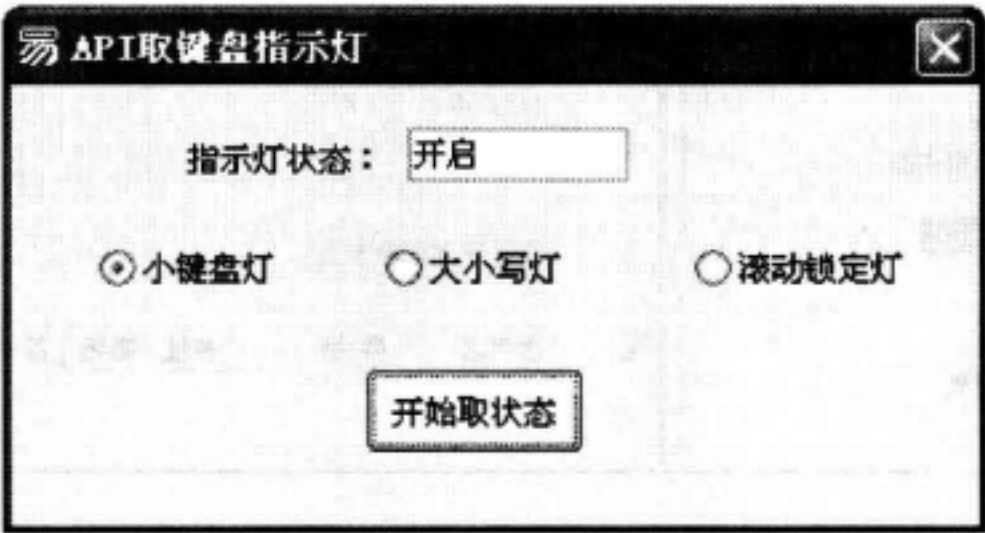


图 16-6 “读取键盘指示灯状态”运行结果

16.3 标准 DLL 库的应用

Windows 系统标准 API 函数被编译到扩展名为“DLL”的文件,随着操作系统的安装而来的,在 Windows2000 以上系统存放在“WinNT\System32”目录中,编程人员在应用程序的开发中,可以调用 Windows 自带的 API 函数。

【范例 16-2】例程 16-2 通过两个 DLL 命令实现取任意窗口(控件)标题的功能。

添加两个 API 函数到易语言 DLL 命令中,一个是“GetWindowText”,一个是“GetWindowTextLength”,都在“user32.dll”函数库中。

通过查找相关资料,了解到“GetWindowText”的功能是“取得一个窗体的标题文字,或者一个控件的内容”;“GetWindowTextLength”的功能是“取得窗口标题文字或控件内容的长度”。完成后的 DLL 命令如图 16-7 所示。

Dll命令名	返回值类型	公开	备 注	
取窗口标题_	整数型		, 取得一个窗体的标题 (caption) 文字, 或者一个控件的内容 (在vb里使用: 使用vb窗体或控件的caption或text属性)	
库文件名:				
user32.dll				
在库中对应命令名:				
GetWindowTextA				
参数名	类 型	传址	数组	备 注
窗口句柄	整数型			欲获取文字的那个窗口的句柄
缓冲区	文本型			预定义的一个缓冲区, 至少有cch+1个字符大小; 随同窗口文字载入
缓冲尺寸	整数型			lp缓冲区的长度:

Dll命令名	返回值类型	公开	备 注	
取标题文本长度_	整数型		,	
库文件名:				
user32.dll				
在库中对应命令名:				
GetWindowTextLengthA				
参数名	类 型	传址	数组	备 注
窗口句柄	整数型			

图 16-7 定义 DLL 命令

具体的程序代码如图 16-8 所示,其中“取窗口标题”子程序实现了通过窗口句柄获得窗口标题的功能。“按钮 1_被单击”子程序主要用来演示取窗口标题的结果。

子程序名	返回值类型	公开	备 注		
取窗口标题	文本型				
参数名	类 型	参考	可空	数组	备 注
窗口句柄	整数型				

变量名	类 型	静态	数组	备 注
标题内容	文本型			
标题长度	整数型			

标题长度 = 取标题文本长度_ (窗口句柄)

标题内容 = 取空白文本 (标题长度)

取窗口标题_ (窗口句柄, 标题内容, 标题长度 + 1)

返回 (标题内容)

子程序名	返回值类型	公开	备 注
_按钮1_被单击			

信息框 (取窗口标题 (编辑框1.取窗口句柄 0), 0, “取窗口标题——编辑框内容”)

图 16-8 “取窗口标题”代码

程序的执行效果如图 16-9 所示。

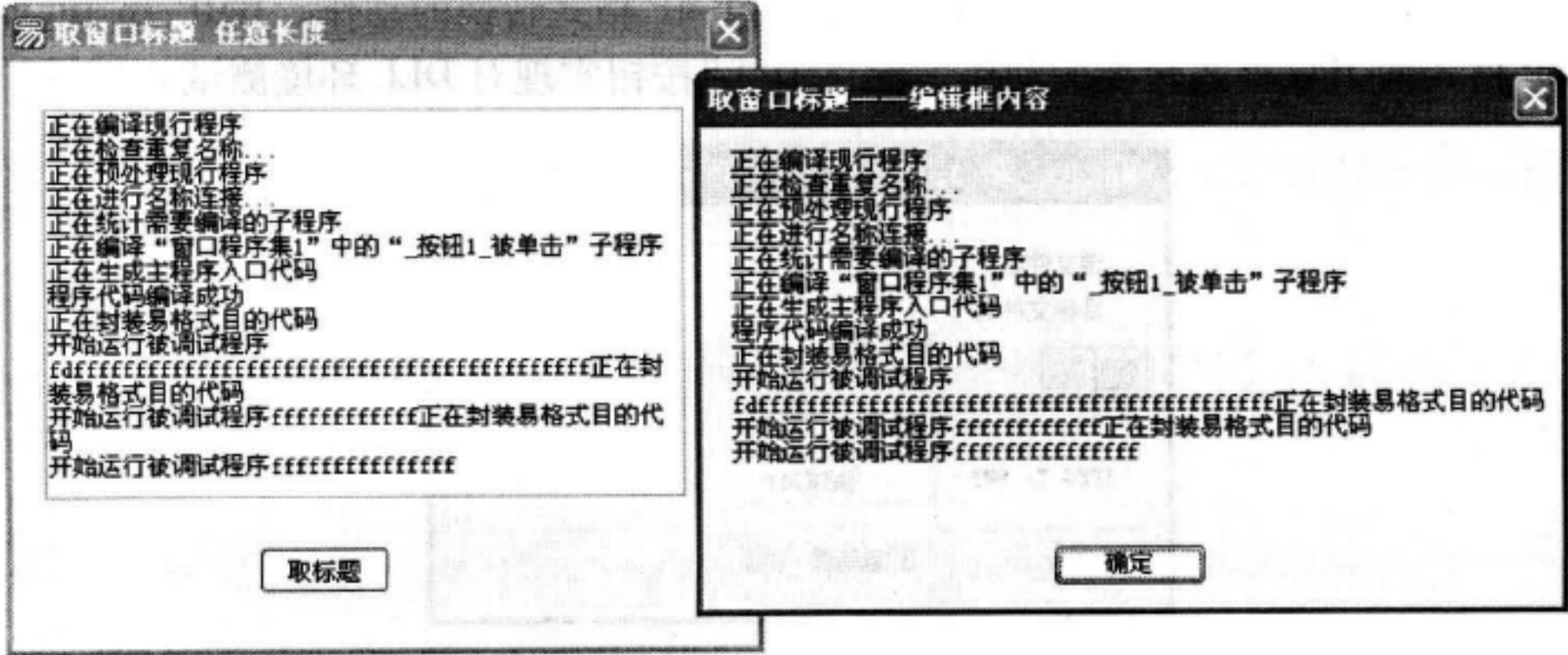


图 16-9 “取窗口标题”运行结果

16.4 外部 DLL 库的应用

电脑技术突飞猛进的发展,一些电脑公司或者计算机编程人员也编制了不少功能强大、使用简便的 DLL 函数库,只要获得了这些函数库,并且了解了函数库中函数的定义形态,其他编程人员也可以调用这些函数。这些函数就称为外部 API 或外部 DLL,在使用这些函数的时候,要先将相应的 DLL 库文件拷贝到系统制定的目录中,或者拷贝到应用程序所在的目录中。

下面通过一个例子来了解外部 DLL 的调用方法。

【范例 16-3】例程 16-3 实现 bmp 图片格式与 jpg 图片格式之间的相互转换。

用到了一个外部函数库,文件名为“Project2. dll”,其中有三个 DLL 命令,“BmpToJpg”实现 BMP 格式图像向 JPG 格式图像转换,“JpgToBmp”实现 JPG 格式图像向 BNP 格式图像转换,“TestDll”实现对 DLL 环境的测试。

首先,在易语言中完成 DLL 命令定义,对“BmpToJpg”命令的对应如图 16-10 所示,可以看到 BMP 格式向 JPG 格式转换时,要对图像的转换品质进行设定,在本例中,使用一个滑块条控件实现。

对“JpgToBmp”命令的对应如图 16-11 所示。

Dll命令名	返回值类型	公开	备 注	
BmpToJpg	整数型			
库文件名：				
project2. dll				
在库中对应命令名：				
BMP2JPG				
参数名	类 型	传址	数组	备 注
BMP文件名	文本型			
JPG文件名	文本型			
图像品质	整数型			

图 16-10 定义“BmpToJpg”命令

Dll命令名	返回值类型	公开	备 注	
JpgToBmp	整数型			
库文件名：				
project2. dll				
在库中对应命令名：				
JPG2BMP				
参数名	类 型	传址	数组	备 注
JPG文件名	文本型			
BMP文件名	文本型			

图 16-11 定义“JpgToBmp”命令

程序窗口界面设计如图 16-12 所示,缺省时滑块条的位置为“70%”,使用两个编辑框来保存图像源文件名称和目标文件名称,使用两个“通用对话框”控件设置打开文件名称和保存文件名称。“JPEG To BMP”按钮实现 JPG 格式向 BMP 格式转换的动作,“BMP To JPEG”按钮实现 BMP 格式向 JPG 格式转换的动作,“测试 DLL”按钮实现对 DLL 环境测试。

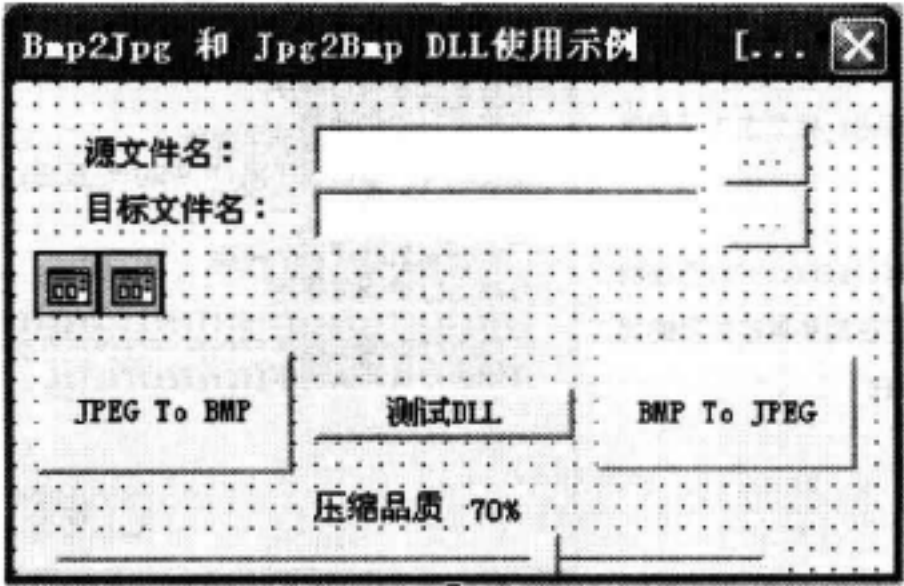


图 16-12 BmpToJpg 和 JpgToBmp DLL 使用示例

读入图像源文件名称的代码如图 16 - 13 所示。这里对读入的文件名称进行判断,自动识别图像格式(BMP 和 JPG 格式之一)。

子程序名		返回值类型	公开	备注
_按钮_取源文件名_被单击				
变量名	类型	静态	数组	备注
文本位置	整数型			
如果真 (通用对话框_打开文件.打开 ())				
编辑框_源文件名.内容 = 通用对话框_打开文件.文件名				
编辑框_目标文件名.内容 = ""				
文本位置 = 寻找文本 (编辑框_源文件名.内容, " ", , 假)				
源文件后缀 = 到大写 (取文本右边 (编辑框_源文件名.内容, 取文本长度 (编辑框_源文件名.内容) - 文本位置))				
如果 (源文件后缀 = "BMP")				
目标文件默认后缀 = "JPG"				
目标文件默认后缀 = "BMP"				
文本位置 = 寻找文本 (通用对话框_保存文件.过滤器, " ", , 假)				
通用对话框_保存文件.过滤器 = 取文本左边 (通用对话框_保存文件.过滤器, 文本位置) + "*" + 目标文件默认后缀				
通用对话框_保存文件.默认文件后缀 = 目标文件默认后缀				

图 16 - 13 “读入图像源文件名称”代码

读入图像目标文件名称的代码如图 16 - 14 所示,目标图像文件的扩展名是自动添加的。

子程序名	返回值类型	公开	备注
_按钮_取目标文件名_被单击			
如果真 (通用对话框_保存文件.默认文件后缀 = "")			
信息框 ("请先选择图像源文件", 0, "出错了")			
返回 0			
如果真 (通用对话框_保存文件.打开 ())			
编辑框_目标文件名.内容 = 通用对话框_保存文件.文件名			

图 16 - 14 “读入图像目标文件名称”代码

实现 JPG 格式向 BMP 格式转换的代码如图 16 - 15 所示,增加了源文件格式出错的识别。

子程序名	返回值类型	公开	备注
_按钮_JPGtoBMP_被单击			
如果真 (源文件后缀 = "BMP")			
信息框 ("图像源文件不是JPG格式,请重新选择", 0, "出错了")			
返回 0			
如果真 (JpgToBmp (编辑框_源文件名.内容, 编辑框_目标文件名.内容) = 1)			
信息框 ("转化成功!", 0,)			

图 16 - 15 “JPG 向 BMP 转换”代码

实现 BMP 格式向 JPG 格式转换的代码如图 16 - 16 所示,增加了源文件格式出错的识别。

子程序名	返回值类型	公开	备注
_按钮_BMPtoJPG_被单击			
如果真 (源文件后缀 ≠ "BMP")			
信息框 ("图像源文件不是BMP格式,请重新选择", 0, "出错了")			
返回 0			
如果真 (BmpToJpg (编辑框_源文件名.内容, 编辑框_目标文件名.内容, 滑块条_压缩品质.位置) = 1)			
信息框 ("转化成功!", 0,)			

图 16 - 16 “BMP 向 JPG 转换”代码

第 17 章 易 DLL 编写及调用

17.1 易 DLL、易模块、API 之间的关系

17.1.1 DLL 基本概念

DLL 名叫动态链接库是英文 Dynamic Link Libray 的缩写,是一个包含共享函数的二进制文件,是 Windows 系统的主要组成部分,它可以同时被多个应用程序调用,是一个只有在运行时才进行连接并共享的函数库。

当应用程序调用 DLL 时候,DLL 被装载到内存中,操作系统会自动在以下位置查找 DLL 库文件。

- 应用程序所在目录
- Windows 系统目录
- Windows 目录
- 系统环境变量(PATH 等)中指定的目录

如果在上述位置找不到对应 DLL 库文件,程序运行就会出错。

17.1.2 易 DLL 与易模块、API 间的关系

易 DLL,顾名思义,指的是用易语言编写出来的 DLL 函数库在其中可以封装一个或多个用易语言环境编写的函数,它既可以由易语言环境调用,也可以由第三方开发环境调用,可以实现多人协作开发较大项目。

在第 15 章介绍了易模块的相关知识,在第 16 章介绍了 API 函数的相关知识,那么三者之间是有什么共同点,又有什么区别呢?

三者的共同点是,它们都是以文件的形式存在磁盘上,在需要的时候,按照某种规则像调用子程序一样,调用其中某个功能(函数)。

三者的不同点是,易模块是由易语言环境编写的,其包含的函数只能在易语言环境中调用,无法供其他编程环境调用;易 DLL 也是由易语言环境编写的,但其包含的函数可以在易语言环境中调用,也可以由第三方开发环境(如 VC、VB 等)来调用,易 DLL 在对中文的处理上有着较强的优势;API 函数则是由 Windows 系统提供的,或者由第三方开发环境编程实现的,其函数既可以由易语言环境来调用,也可以由第三方开发环境来调用。

本章节主要介绍易 DLL 的编写及在易语言环境下的调用方法,对第三方开发环境的调用方法,重点介绍注意事项。

17.2 易 DLL 的开发与编译

易语言从 3.6 版本开始,就能够支持对 DLL 动态链接库的开发,编译出标准的 WIN32

DLL 文件,供其他编程环境调用。

17.2.1 进入 DLL 编写环境

有两种方法来建立动态链接库编写环境,一是在启动易语言时,在“新建”窗口中选择“Windows 动态链接库”进入,二是在易语言环境中,点击菜单“新建”命令,或点击工具条“新建”按钮,在弹出的“新建”窗口中选择“Windows 动态链接库”进入,如图 17-1 所示。

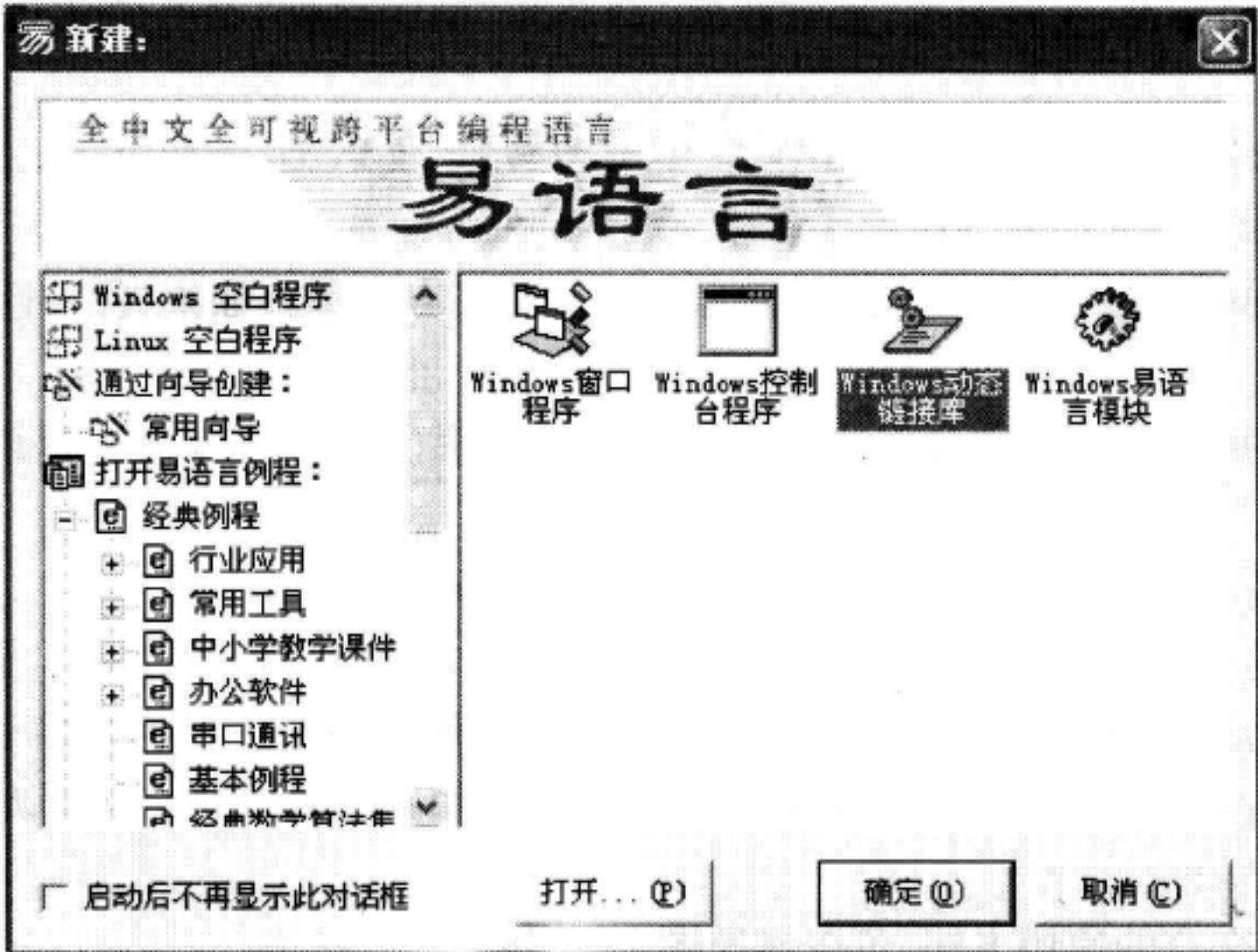


图 17-1 “新建”窗口选择“Windows 动态链接库”

进入易语言编写动态链接库的代码设计工作区(程序集)如图 17-2 所示。

程序集名	保留	保留	备注
程序集1			

子程序名	返回值类型	公开	备注
_启动子程序	整数型		请在本子程序中放置动态链接库初始化代码

_临时子程序 0 在初始化代码执行完毕后调用测试代码
返回 0 返回值被忽略。

子程序名	返回值类型	公开	备注
_临时子程序			

本名称子程序用作测试程序用,仅在开发及调试环境中有效,编译发布程序前将被系统参数及返回值类型。

图 17-2 动态链接库的代码设计工作区

17.2.2 编写 DLL 源代码

易语言中 DLL 的编写过程与前面提到的易模块、易子程序类似,其中需要公开、允许其他环境调用的函数需要有对外的公开接口,并且只有公开的子程序才能被外部环境调用。

编写 DLL 的公开接口只能由以下方式产生,新建一个子程序,然后把其“公开”属性勾选即可。在 DLL 中任何程序集中选中“公开”属性的子程序都可以作为一个对外接口被其他外部环境调用。建立公开接口的方法如图 17-3 所示。

下面通过一个例子来创建一个 DLL 库源码文件。

子程序名	返回值类型	公开	备注
子程序1		✓	

图 17-3 勾选“公开”属性建立公开接口

例 17-1 是一个对磁盘文件按照 RC4 算法进行加解密的接口源程序代码,主要包含一个公开的加密子程序和解密子程序。

文件内容的加解密,主要通过“加密数据”命令和“解密数据”命令来实现,具体命令的解释如下。

调用格式:〈字节集〉加密数据(字节集 字节集数据,文本型 密码文本,[整数型 加密算法]) - 数据操作支持库 - > 数据加解密

命令功能:加密一段字节集数据,返回加密后的结果字节集。如果失败,返回空字节集。

参数说明:

参数<1>的名称为“字节集数据”,类型为“字节集(bin)”。为命令提供所需的字节集数据。

参数<2>的名称为“密码文本”,类型为“文本型(text)”。

参数<3>的名称为“加密算法”,类型为“整数型(int)”,可以被省略。指定具体使用的加密算法,可以为以下常量值之一:1: #DES 算法; 2: #RC4 算法。加密和解密必须使用相同的算法,有关算法的具体说明请参阅有关文献。如果本参数被省略,则默认值为 1,即 DES 算法。

“加密数据”命令的使用方法,可以参照图 17-4。

调用格式:〈字节集〉解密数据(字节集 字节集数据,文本型 密码文本,[整数型 加密算法])。

命令功能:解密一段加密后的字节集数据,返回解密后的结果字节集。注意本命令并不对密码文本进行校验,如果密码提供错误,将返回错误的结果。如果失败,返回空字节集。

参数说明:

参数<1>的名称为“字节集数据”,类型为“字节集(bin)”。为命令提供所需的字节集数据。

参数<2>的名称为“密码文本”,类型为“文本型(text)”。

参数<3>的名称为“加密算法”,类型为“整数型(int)”,可以被省略。指定具体使用的加密算法,可以为以下常量值之一:1: #DES 算法; 2: #RC4 算法。加密和解密必须使用相同的算法,有关算法的具体说明请参阅有关文献。如果本参数被省略,则默认值为 1,即 DES 算法。

“解密数据”命令的使用方法,可以参照图 17-5。

17.2.3 编译易 DLL

只有使用正版软件才能进行编译,所以在编译前,需要将加密狗插入计算机接口,待正确识别后再进行编译。

点选“编译→静态编译”菜单命令,或者使用热键“Shift + F7”,可以弹出窗口如图 17-6 所示。

输入要生成到的 DLL 文件名称,如本例“17-1”,点击“保存”按钮,则会在“状态夹”中的“输出”子夹会有编译过程的提示,如图 17-7 所示信息表示编译正确完成。

子程序名	返回值类型	公开	备注		
jiamiwenjian	逻辑型	✓			
参数名	类型	参考	可空	数组	备注
filename	整数型				
str	整数型				

变量名	类型	静态	数组	备注
文件号	整数型			
文件长度	整数型			
文件内容	字节集			
成功	逻辑型			

文件号 = 打开文件 (指针到文本 (filename), ,)
文件长度 = 取文件长度 (文件号)
移到文件首 (文件号)
文件内容 = 读入字节集 (文件号, 文件长度)
文件内容 = 加密数据 (文件内容, 指针到文本 (str), 2)
移到文件首 (文件号)
成功 = 写出字节集 (文件号, 文件内容)
关闭文件 (文件号)
返回 (成功)

图 17-4 加密文件子程序代码

子程序名	返回值类型	公开	备注		
jiemiwenjian	逻辑型	✓			
参数名	类型	参考	可空	数组	备注
filename	整数型				
str	整数型				

变量名	类型	静态	数组	备注
文件号	整数型			
文件长度	整数型			
文件内容	字节集			
成功	逻辑型			

文件号 = 打开文件 (指针到文本 (filename), ,)
文件长度 = 取文件长度 (文件号)
移到文件首 (文件号)
文件内容 = 读入字节集 (文件号, 文件长度)
文件内容 = 解密数据 (文件内容, 指针到文本 (str), 2)
移到文件首 (文件号)
成功 = 写出字节集 (文件号, 文件内容)
关闭文件 (文件号)
返回 (成功)

图 17-5 解密文件子程序代码

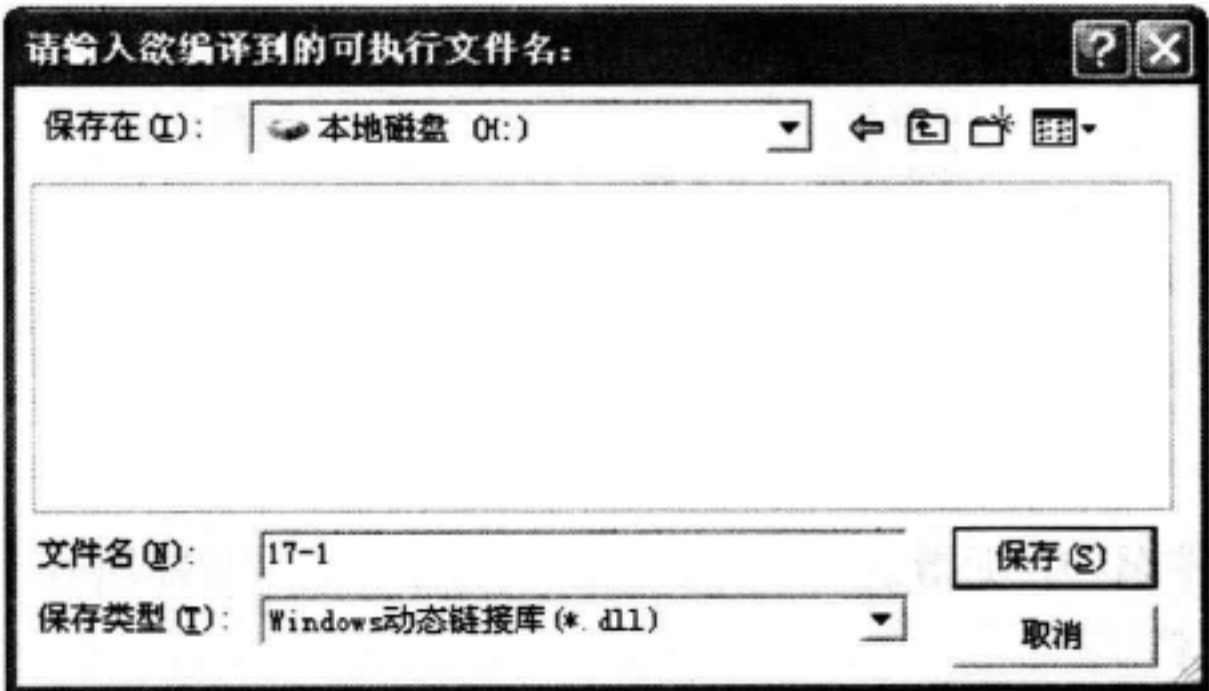


图 17-6 “输入编译到的可执行文件名”窗口

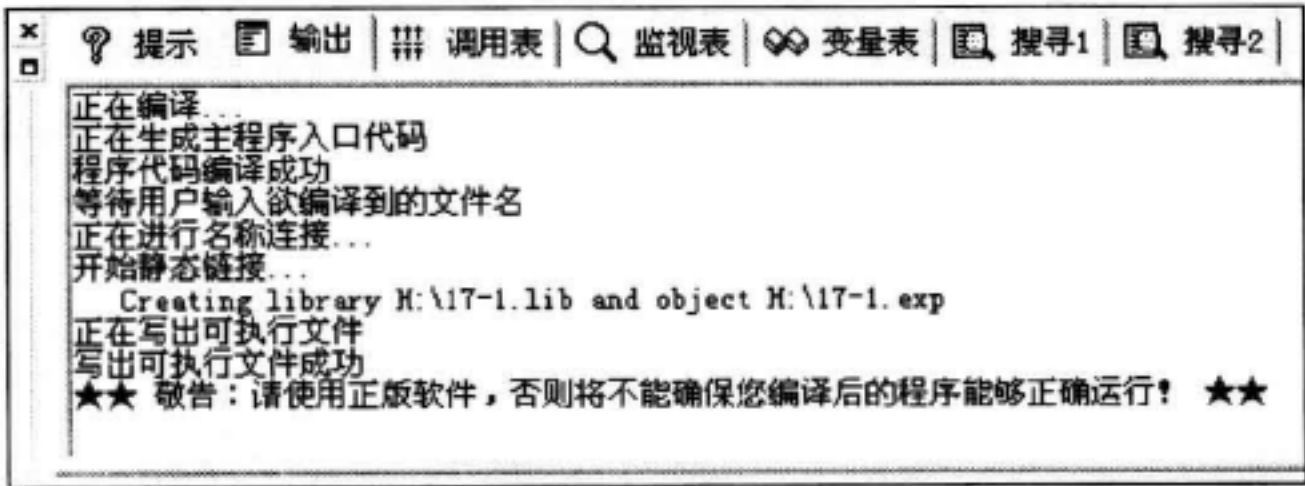


图 17-7 “状态夹”输出正确编译信息

在磁盘上能够看到多了一个扩展名为“DLL”的文件“17-1.dll”。

如果需要对此 DLL 库文件做一些通用说明,可以参考在易模块一章中介绍的那样在“程序配置对话框”中加以说明,具体方法为点击“程序→配置”,在弹出的“程序配置对话框”中输入必要的信息即可。

17.3 如何调用易 DLL

调用易 DLL 一般分为两种情况,一个是易环境中调用易 DLL,一个是在 VB、VC、Delphi 等

第三方开发环境中调用。本文重点介绍在易环境下的调用。

调用易 DLL 和调用 API 函数的方法是一样的。新建一个“Windows 窗口程序”，选择“插入→DLL 命令”菜单命令，将 DLL 声明按照要求加入即可。如图 17-8 所示。

Dll命令名	返回值类型	公开	备 注	
文件加密	逻辑型			
库文件名：				
17-1.dll				
在库中对应命令名：				
jiamiwenjian				
参数名	类 型	传址	数组	备 注
文件名称	文本型			
文本串	文本型			

Dll命令名	返回值类型	公开	备 注	
文件解密	逻辑型			
库文件名：				
17-1.dll				
在库中对应命令名：				
jiemiwenjian				
参数名	类 型	传址	数组	备 注
文件名称	文本型			
文本串	文本型			

图 17-8 添加 DLL 声明

【注意】

- (1) “DLL 库文件名”中所指的 DLL 文件必须位于 EXE 文件所在目录或系统目录，否则在程序运行时将会出错。
- (2) 调用 DLL 时 DLL 对应命令名必须和接口名称完全一样。

例程 17-2 演示了如何在易环境下调用易 DLL 命令的方法，其中调用“文件加密”函数的方法如图 17-9 所示。

子程序名	返回值类型	公开	备注	
按钮1_被单击				

变量名	类型	静态	数组	备注
文件名称	文本型			
文本串	文本型			

文件名称 = “d:\1.txt”

文本串 = “rqe237482tr843rcj43nr43980u5v34j36b45658m698,0n4vrtertbtv3t3v45yb34b53t64nb76567n”

如果真 (文件加密 (文件名称, 文本串))

信息框 (“加密成功”, 0,)

图 17-9 调用“文件加密”函数的方法

【注意】在易 DLL 中的接口函数的参数必须是基本数据类型，也可以是文本型，但不能为字节集型。

那么易语言在进行字节集参数传递的时候要如何处理呢？易语言考虑到了这个问题，提供了 3 个命令：“指针到文本”、“指针到字节集”、“写到内存”。当要传递参数是文本型或者字

节集型时,易语言会自动将数据转换成指针进行传递,待需要的时候再使用“指针到文本”或“指针到字节集”把原来的文本或字节集返回出来。如果需要返回字节集数据,则需要先通过“写到内存”命令将返回的值写入内存,再将地址易整数型参数返回即可。

例程 17-3 是一个易 DLL 函数中字节集参数传递的例子,磁盘文件 17-3.dll 为编译后的函数库。

演示字节集数据相加的子程序代码如图 17-10 所示。“rr”这个局部变量在程序中没有实际意义,它的作用是通过赋值来分配一块地址空间,这样就可以通过“取变量数据地址”命令来获得地址,通过“写到内存”命令来将返回的字节集数据写到内存块中。

子程序名	返回值类型	公开	备 注		
zijiejijiafa	整数型	✓			
参数名	类 型	参 考	可 空	数 组	备 注
被加数字节集	整数型				接收的是被加数字节集数据地址
被加数字节集长度	整数型				
加数字节集	整数型				接收的是加数字节集数据地址
加数字节集长度	整数型				
变量名	类 型	静 态	数 组	备 注	
内存指针	整数型				
rr	字节集				

’ 需要先给变量赋值,生成地址

rr = { 11 } ’ 给变量赋值

内存指针 = 取变量数据地址 (rr) ’ 取出变量数据地址

’ 将运算结果写入到内存变量中

写到内存 (指针到字节集 (被加数字节集, 被加数字节集长度) + 指针到字节集 (加数字节集, 加数字节集长度), 内存指针,)

返回 (内存指针) ’ 返回内存变量地址

图 17-10 易 DLL 中函数字节集型参数的传递

例程 17-4 是调用易 DLL 函数的例子。其中 DLL 函数的定义如图 17-11 所示。

Dll命令名	返回值类型	公开	备 注	
jiafa	整数型			
库文件名：				
17-3.dll				
在库中对应命令名：				
zijiejijiafa				
参数名	类 型	传址	数组	备 注
a	字节集			
b	整数型			
c	字节集			
d	整数型			

图 17-11 字节集相加 DLL 函数的定义

从图 17-11 可以看到在参数的定义上,变量“a”和“c”都定义为“字节集”类型,与原接口子程序定义的“整数型”类型不符,这是允许的,易语言在调用的时候,会自动将字节集型数据的地址传递到接口子程序中,而不是将数据本身传递到接口子程序中,这样就满足了接口子程序的参数要求,而不会出错。也省去了一个取字节集数据地址的过程。

调用字节集相加函数的方法如图 17-12 所示。

例程 17-4 的执行效果如图 17-13 所示,可以看到将两个字节集串相加并显示出来了。

易 DLL 函数也可以被 VB、VC、Delphi 等其他开发环境调用,这样易语言开发者就可以与

子程序名	返回值类型	公开	备注
按钮1_被单击			

变量名	类型	静态	数组	备注
a	字节集			
b	字节集			

a = 到字节集 ("xwj")
b = 到字节集 ("0001")
信息框 (到文本 (指针到字节集 (jiafa (a, 取字节集长度 (a), b, 取字节集长度 (b)), 取字节集长度 (a) + 取字节集长度 (b))), 0,)

图 17-12 演示调用字节集相加函数的方法

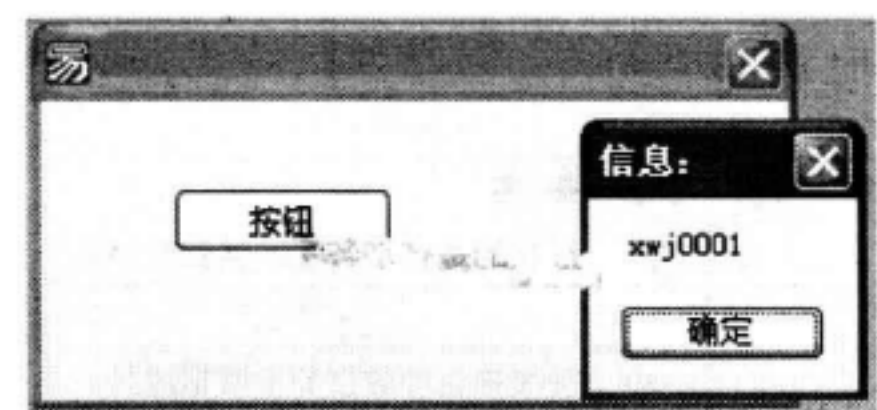


图 17-13 字节集相加执行效果

其他环境的开发者一道合作,共同开发较大的项目,但是由于易环境主要偏重于中文习惯处理数据,因此在合作的过程中要注意由于数据类型的差异造成的冲突。

在易 DLL 函数开发中,涉及基本数据类型,在调用时没有什么问题,但是在涉及文本/字节集等类型时,在第三方开发环境,如 Delphi,如果采用静态调用 DLL 函数的方式来调用易 DLL 函数,会导致程序退出时发生异常,但是如果采用动态调用 DLL 函数的方式来调用易 DLL 函数,则一切正常。

【注意】在与第三方环境合作时,建议调用 DLL 函数时,采用动态调用为好。